Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Algorithms 53 (2004) 137–168

Journal of Algorithms

www.elsevier.com/locate/jalgor

Analysis of queueing policies in QoS switches ☆

An Zhu [1]

Department of Computer Science, Stanford University, Stanford, CA 94305, USA

Received 13 October 2002

Available online 5 June 2004

Abstract

It is widely accepted that next-generation networks will provide guaranteed services, in contrast to the "best effort" approach today. We study and analyze queueing policies for network switches that support the QoS (Quality of Service) feature. One realization of the QoS feature is that packets are not necessarily all equal, with some having higher priorities than the others. We model this situation by assigning an intrinsic value to each packet. In this paper we are concerned with three different queueing policies: the *nonpreemptive* model, the *FIFO preemptive* model, and the *bounded delay* model. We concentrate on the situation where the incoming traffic overloads the queue, resulting in packet loss. The objective is to maximize the total value of packets transmitted by the queueing policy. The difficulty lies in the unpredictable nature of the future packet arrivals. We analyze the performance of the online queueing policies via competitive analysis, providing upper and lower bounds for the competitive ratios. We develop practical yet sophisticated online algorithms (queueing policies) for the three queueing models. The algorithms in many cases have provably optimal worst-case bounds. For the nonpreemptive model, we devise an optimal online algorithm for the common 2-value model. We provide a tight logarithmic bound for the general nonpreemptive model. For the FIFO preemptive model, we improve the general lower bound to 1.414, while showing a tight bound of 1.434 for the special case of queue size 2. We prove that the bounded delay model with uniform delay 2 is equivalent to a modified FIFO preemptive model with queue size 2. We then give improved upper and lower bounds on the 2-uniform bounded delay model. We also show an improved lower bound of 1.618 for the 2-variable bounded delay model, matching the previously known upper bound.
© 2004 Elsevier Inc. All rights reserved.

☆ A preliminary version of this paper was merged with another paper by Nir Andelman and Yishay Mansour, which appeared in 14th Annual SIAM–ACM Symposium on Discrete Algorithms (SODA), 2003.

E-mail address: anzhu@cs.stanford.edu.

[1] Supported by a GRPW fellowship from Bell Labs, Lucent Technologies.

0196-6774/$ – see front matter © 2004 Elsevier Inc. All rights reserved.
doi:10.1016/j.jalgor.2004.04.007

1. Introduction

Currently, the internet infrastructure employs “best effort” policy for all traffic streams, providing no guarantee on packet delivery. The uncertainty of its performance is not satisfactory for many network applications. The widely foreseen next-generation networks will provide guaranteed services to meet various user demands. This gives rise to the recent interest in the Quality of Service (QoS) feature.

This vision has been around the networking community for more than a decade [10]. For instance, ATM networks serve as an example of a unified architecture that supports a diverse set of service classes. Of late, there has been tremendous interest in IP in providing *differentiated services* via QoS guarantees. The basic methodology of QoS is rather intuitive—committing resources to each admitted connection. Thus, the network is capable of providing different users with different classes of service. In particular, a contract between users and service providers ensures that the network maintains the performance guarantees provided the users stick to their commitments about traffic generation.

However, due to a variety of reasons, the incoming traffic patterns may not coincide with that specified in the service contract. A typical situation is that the traffic from the user does not conform to the patterns defined in the contract. The difficult situation is when the traffic exceeds the allocated bandwidth at some point. Another equally serious problem is that by guaranteeing the worst-case performance, the QoS network might not be efficient due to its conservative policy, as network traffic tends to be bursty. Recognizing this phenomenon, most modern QoS networks allow some “overbooking,” employing the policy popularly known as *statistical multiplexing* [4]. In either case, QoS networks must resolve the unavoidable issue of overloading. This paper analyzes the performance of queueing policies under overloading situations using competitive analysis.

In the past few years the networking community has had an increasing interest in QoS networks [3,7–9]. A major new paradigm suggested is the assured service [2]. This service has a loose guarantee in which traffic conforming to the specified pattern is much less likely to be interrupted in the network. This approach leads to two types of packets in the system: those of high priority (conformed traffic) and those of low priority (unconformed traffic). High priority packets stands less chance of being dropped by the network.

We abstract the above problem as follows: We assume a value associated with each packet: value 1 for the low priority packets, and value $\alpha > 1$ for the high priority packets. This is called the 2-*value model*. For differing network requirements, we can adjust α to achieve the desired performance guarantee. We also consider the extension where packets take on arbitrary values in the range $[1, \alpha]$. We assume that the queue can hold up to B packets. The goal is to maximize the total value of the packets transmitted. In terms of competitive analysis, we compare the total value of the packets transmitted by an online algorithm to that of the optimal offline algorithm. We say that an online algorithm has a competitive ratio of β, if for any packet arrival sequence, the total value transmitted is at least $1/\beta$ fraction of that of the optimal. This paper provides upper and lower bounds for the competitive ratios on three different queueing policies. The *nonpreemptive policy* transmits all packets admitted into the queue; observe, under this policy, the queue can easily maintain a FIFO order. The *FIFO preemptive policy* is allowed to drop packets already admitted to the queue. The *bounded delay policy*, on the other hand, transmits packets in any order, but each packet must be transmitted before a fixed deadline.

The nonpreemptive policy for the 2-value model was first proposed by Aiello et al. [1], who studied four different queueing policies. Each of these four policies has competitive ratio strictly worse than the known lower bound of $(2\alpha - 1)/\alpha$. We present a practical online algorithm and prove that its competitive ratio is precisely $(2\alpha - 1)/\alpha$, thereby completely solving the problem for this case. For the general model, where the packet values lie in the range $[1, \alpha]$, we establish matching upper and lower bounds of $\Theta(\log \alpha)$.

The FIFO preemptive policy has been studied extensively. The 2-value model was considered by Kesselman and Mansour [5], who provided approximately tight bounds for large values of α and B. We concentrate on the general model, of which previous lower and upper bounds were developed by Kesselman et al. [4] and Kesselman et al. [6]. In particular, the lower bound is 1.281, and the upper bound is 1.983. We establish tight upper and lower bounds of $(5 + \sqrt{13})/6 \approx 1.434$ for $B = 2$. As a byproduct of our techniques, we improve the lower bound for general queue sizes to $\sqrt{2} \approx 1.414$.

Our techniques for the FIFO preemptive policy apply to the bounded delay policy as well. In particular, we show that the modified FIFO preemptive model with queue size 2 is equivalent to the 2-uniform bounded delay model, where each arriving packet must be transmitted within the next 2 time units. We establish upper and lower bounds of 1.414 and 1.366, respectively. This is an improvement upon the previous bounds of 1.434 and 1.25, respectively, due to Kesselman et al. [4]. For the model where some of the arriving packets must be sent out within one time unit, namely, the 2-variable bounded delay model, we establish a lower bound of 1.618. This is an improvement of the previous lower bound of 1.414 and matches the upper bound due to Kesselman et al. [4].

The rest of the paper is organized as follows, Section 2 gives a tight analysis for the nonpreemptive model, Section 3 deals with the FIFO preemptive model, Section 4 presents results for the bounded delay model, finally Section 5 concludes with open problems.

2. Nonpreemptive queueing policy

We first consider the 2-value nonpreemptive model. Consider a switch buffer (queue) with enough memory to hold B packets (or B slots). Packets have identical size and occupy one slot each. A low priority packet has benefit 1, while a high priority packet has benefit $\alpha > 1$. Upon the arrival of a packet, the queueing policy (online algorithm) has to decide immediately whether to accept the packet in the queue, or reject it. Rejected packets are lost forever. Accepted packets stay in the queue, and get transmitted at a rate of 1 packet per time unit. When a queue is full (i.e., contains B packets), no more packets can be accepted, until some packets are transmitted to free up the queue space. For convenience, we assume that packets are transmitted at integral times, i.e., at each integral time, the number of packets in the queue (if nonzero) goes down by one. In addition, no two packets arrive at the same time and no packet arrives at integral times.[2] The aim of the queueing policy is

[2] These constrains are only added to make the analysis simpler to present and follow.

to maximize the sum of the benefits of all packets that get transmitted. For convenience, time starts at 0. We use j and its related forms j' etc. to denote integral times, use t and its related forms t' etc. to denote nonspecific times.

Aiello et al. [1] showed that for a particular value of α, there is a general lower bound for any type of online algorithms. For the sake of completeness, we provide the proof below. From now on, we use *OPT* to denote the optimal offline algorithm. And for convenience, we use H.P. to denote "high priority," and L.P. to denote "low priority."

Theorem 1. *Any online policy* (*deterministic or randomized*) *has a competitive ratio of at least* $(2\alpha - 1)/\alpha$.

Proof. At time t (where $0 < t < 1$), suppose B L.P. packets arrive one after another, infinitesimally apart from each other.[3] Let x denote the expected number of L.P. packets the online algorithm accepts. If $x \leqslant (\alpha/(2\alpha - 1))B$, then the adversary does not release any new packets. *OPT* would have accepted and transmitted all B packets. And so the online algorithm transmits no more than $\alpha/(2\alpha - 1)$ fraction of the packets. Otherwise, $x > (\alpha/(2\alpha - 1))B$, and the adversary will release B H.P. packets at time t' (where $t < t' < 1$). *OPT* then would have rejected all the L.P. packets, and accepted all B H.P. packets. The online algorithm at best can accept an additional $B - x$ H.P. packets before the queue becomes full. It follows that the competitive ratio is at least

$$\frac{B\alpha}{x + (B - x)\alpha} \geqslant \frac{(2\alpha - 1)\alpha}{\alpha + (\alpha - 1)\alpha} = \frac{2\alpha - 1}{\alpha}. \qquad \square$$

We now present the Ratio Partition policy, which builds on early policies given in [1], and show that it's competitive ratio is indeed $(2\alpha - 1)/\alpha$.

We use *RP* to denote the Ratio Partition algorithm. *RP* always accepts H.P. packets as long as the queue is not full (i.e., there is free slot), and conditionally accepts L.P. packets, guaranteeing that there is a good mix of H.P. and L.P. packets in the queue. We introduce the concept of "matching." L.P. packets are either matched with H.P. packets, or unmatched. Initially, *RP* accepts a L.P. packets as unmatched. An accepted H.P. packet is matched up to $\alpha/(\alpha - 1)$ unmatched L.P. packets currently in the queue, starting with the earliest. In the case of $\alpha/(\alpha - 1)$ not integral, we allow the number of matched L.P. packets to be fractional, since matching is only conceptual. We define the "threshold ratio" of the queue as the total number of unmatched L.P. packets vs. the total number of free slots in the queue. When a L.P. packet arrives, *RP* accepts only if afterwards the threshold ratio is at most $\gamma = \alpha/(\alpha - 1)$.[4] We call γ the threshold parameter.

We illustrate the algorithm using a concrete example. Let's suppose $B = 12$, $\alpha = 2$, and initially the queue is empty. The threshold parameter γ is then $\alpha/(\alpha - 1) = 2$ in this case. Now 6 L.P. packets arrive, *RP* will be able to accept all 6 of them, because the threshold

[3] For the rest of the paper, we will simply state "B numbers of packets arrive," with the understanding that they arrive one after another, infinitesimally apart.

[4] Notice that inevitably, we will encounter some roundoff errors due to the inability to accept partially an L.P. packet, but this is minor, and not considered here.

ratio afterwards is $6/6 = 1 < 2$ (the queue contains 6 unmatched L.P. packets and 6 free slots). Then two H.P. packets arrive, *RP* accepts both. Now each H.P. packet is matched up to 2 unmatched L.P. packets in the queue. So the queue content is now 4 matched L.P. packets, followed by 2 unmatched L.P. packets, followed by 2 H.P. packets, and finally 4 free slots. Then suppose 4 more L.P. packets arrive, *RP* will only accept the first two, because afterwards the threshold value is $4/2 = 2$. Any additional acceptance of the L.P. packet will raise the threshold ratio above 2. But after the queue transmits some packets, more free slots are introduced, or some unmatched L.P. packets have departed, *RP* will be able to accept more L.P. packets.

We now turn to the competitive analysis of *RP*. Before analyzing *RP*, we first bring more structure to *OPT* without changing its optimality.

Lemma 1. *We can freely exchange any two packets' positions in OPT's queue at any time, without changing the total value of the packets transmitted by OPT.*

Proof. Since every packet accepted to the queue is always transmitted, it doesn't matter in which order they are sent in terms of total value transmitted. In fact, the total value of the packets transmitted is the same as the total value of the packets accepted. Thus in the analysis below, we sometimes argue in terms of packets accepted, whenever convenient. □

Lemma 2. *If OPT is idle at time j, then any online algorithm must also be idle at time j.*

Proof. Suppose the contrary situation occurs, we look at the first such time j when *OPT* is idle and the online algorithm is not. That means at time j, *OPT*'s queue is empty, and the online algorithm's queue contains at least one packet to be sent. We now look backwards in time, and find the last time $t' < j$ when *OPT* has fewer number of packets in its queue compared to that of the online algorithm, i.e., during time $[t', j]$ *OPT* always has fewer number of packets in its queue. Since no two packets can arrive at exactly the same time, that means the number of packets in a queue changes only in unity. Also notice that the number of packets decreases synchronously for both *OPT* and the online algorithm. So at a time infinitesimally before t', *OPT* and the online algorithm have the same number of packets in their queues. Such time t' exists since initially both queues are empty. The only event that can occur at time t' to have caused this sudden change is that the online algorithm accepted a packet while *OPT* rejected the same packet. We then modify *OPT* to accept such a packet. Now during time $[t', j]$, the number of packets in *OPT*'s queue is still upper bounded by that of the online algorithm. So *OPT* never has to reject any packets that it originally accepted during this time period. At time j, *OPT* has at most one packet in its queue to be transmitted. Clearly *OPT* can accept whatever packets accepted originally after time j as well. We have argued that *OPT* can accept one more packet, thus the total value of accepted packets has increased, a contradiction to the optimality of *OPT*. □

Lemma 3. *We can modify OPT so that it accepts all the H.P. packets that RP accepts, without changing its optimality.*

Proof. Because nonpreemptive queueing has a matroid structure, we can construct a particular optimal offline solution as follows [1]: first ignore all L.P. packets, and just greedily accept the H.P. packets as long as the queue is not full. In addition, we accept the L.P. packets that won't interfere with the H.P. packets accepted. Let OPT^H denote the algorithm that only accepts H.P. packets greedily and ignores the L.P. packets. We claim that the set of packets OPT^H accepted is a superset of the H.P. packets that *RP* accepts. Since all the packets accepted by OPT^H are included in the optimal offline solution we are constructing, we will prove this lemma. We first show that the number of packets in OPT^H's queue is always no more than that of *RP*. We prove this by induction. Assume during time $[0, t)$ the hypothesis is true, now consider time t. If a L.P. packet arrives at time t, since OPT^H won't accept any L.P. packet, the hypothesis is true. If a H.P. packet arrives, *RP* always greedily accepts, unless its queue is full, in which case *RP* has the most number of packets possible in its queue. So afterwards the hypothesis is maintained. If a packet departs from OPT^H's queue at time t, by the inductive hypothesis, a packet must also depart from *RP*'s queue. If a packet departs from *RP*'s queue and vice versa, we are again fine. Otherwise OPT^H is idle, which means its queue is empty, the smallest possible. So the hypothesis is maintained after any of the departing situations above. This completes the inductive proof. Thus if *RP* accepts a H.P. packet, since OPT^H has more free slots than *RP*, OPT^H will accept that packet for sure. □

Lemma 4. *We can restrict the sequence of packet arrivals to only contain the packets that either OPT or RP accepted. On the new sequence, OPT and RP accept exactly the same packets as they would on the original sequence.*

Proof. Clearly, *OPT* can accept exactly the same packets throughout, and maintain its optimality by the matroid property. For *RP*, it only rejects H.P. packets if the queue is full, and rejects the L.P. packets if the restriction on the threshold ratio is met. So the packets accepted on the new sequence is going to be the same as well. □

We let S denote this restricted sequence and our analysis will only concentrate on *OPT* and *RP*'s behaviors on S. In particular, from Lemma 3, we know the following.

Corollary 1. *The H.P. packets in S are exactly the ones that OPT accepted.*

We further restrict *OPT*'s behavior as follows.

Lemma 5. *If at time t, RP accepts a L.P. packet p while OPT rejects, and later OPT accepts another L.P. packet p' before its queue is full, then we modify OPT to accept p instead and reject p' later, maintaining the optimality.*

Proof. We look into the future $t' \geqslant t$, where *OPT*'s queue is full for the first time since t. *OPT* exchanges the two packets p and p'. Since the queue was never full before t', the exchange will guarantee that the modified *OPT*'s queue is never overflown till t', when it contains the same number of packets as the original one. Obviously, this does not change the value of the optimal solution. Note that if during time $(t, t']$, *OPT* only accepts H.P.

packets and no L.P. packet, accepting p at time t will interfere with the H.P. packets, then *OPT* must reject p. $\square$

We now are ready for the analysis of *RP*. We imagine the packets transmitted by an algorithm being placed on a one-way infinite tape T. We use $T[i]$ to denote the tape position with index i, for $i \geqslant 0$. $T[j]$ holds the packet transmitted at time $j+1$. We can conceptually think an algorithm as simply placing packets onto the tape, instead of accepting packets into the queue. Thus, a packet p arrived between time $(j, j+1)$ can only be placed at a tape position anywhere from $T[j]$ to $T[j+B-1]$. An algorithm would place p in the first such empty tape position, corresponding to accepting p into the queue. We often refer to packet p by its tape position, i.e., $p = T[q]$ for some $j \leqslant q \leqslant j+B-1$. We use $I(p)$ to denote the index of packet p on the tape, i.e., $I(p) = q$. The total value of the packets transmitted is then the total value of the packets placed on the tape. When referring to either *OPT* or *RP*'s tape, we use the superscripts T^{OPT}, T^{RP}, I^{OPT}, I^{RP}, and omit these superscripts wherever the context is clear. We use $T^{OPT}[i, j]$ ($T^{RP}[i, j]$, respectively) to denote the total value of packets from $T[i]$ to $T[j]$ on *OPT*'s (*RP*'s, respectively) tape.

Lemma 6. *At any time, packets that appear at the same positions in OPT and RP's queues occupy the same tape positions, and vice versa.*

Proof. At any time t, if p is the ith packet in *OPT*'s queue, and q is the ith packet in *RP*'s queue ($1 \leqslant i \leqslant B$), then both p and q will be transmitted at time $\lceil t \rceil + i - 1$. So $I(p) = \lceil t \rceil + i - 2$ and $I(q) = \lceil t \rceil + i - 2$. Similarly, packets with index j on the tapes, $T^{OPT}[j]$ and $T^{RP}[j]$, always appear at the same positions together in *OPT* and *RP*'s queues. It could happen that only one of the packets is present in the queue, but when both packets are present, the positions they occupy in the queues are the same. $\square$

We now present some high level ideas of the analysis. Let t be the first time *RP* and *OPT* disagree with each other. We'll pick a suitable future time $t' > t$, and argue that the total value of packets *RP* accepted during time $[0, t']$ is at least a $\alpha/(2\alpha - 1)$ fraction of that of *OPT*. The analysis then continues after time t', finds the next time instance where the two algorithms disagree with each other, find another suitable time t'' further in the future, compare values of packets accepted between time $(t', t'']$, and so on. In the tape world, let $T^{OPT}[u]$ ($T^{RP}[v]$, respectively) and $T^{OPT}[u']$ ($T^{RP}[v']$, respectively) denote the corresponding starting and ending positions of packets accepted by *OPT* (*RP*, respectively) during time $[0, t']$. Since there may be idle times, we simply require that there are no packets accepted by *OPT* (*RP*, respectively) outside of $[0, t']$ in between positions $T^{OPT}[u]$ and $T^{OPT}[u']$ ($T^{RP}[v]$ and $T^{RP}[v']$, respectively). Initially at time 0, we may set $u = v = 0$. The time t' we pick will guarantee that $u' = v'$, i.e., at time t' the numbers of packets in *OPT* and *RP*'s queue are the same again. Thus such relations ($u = v$ and $u' = v'$) can be maintained for all future analysis as well. Conceptually, our analysis breaks a tape into pieces, and compares the tape contents of *OPT* and *RP* between two consecutive break points.

Theorem 2. *We can break down a tape at some selected positions $bp_1 = 0 < bp_2 < bp_3 < \cdots$, at times $t_1 = 0 < t_2 < t_3 < \cdots$, such that*

$$\frac{T^{OPT}[bp_i, bp_{i+1} - 1]}{T^{RP}[bp_i, bp_{i+1} - 1]} \leqslant C, \quad \textit{where } C = \frac{2\alpha - 1}{\alpha}.$$

Proof. We use induction on the following inductive hypothesis: at time t_i, the following conditions hold:

(a) The numbers of packets in *OPT* and *RP*'s queue are the same.
(b) The queue contents for packets with indices bp_i or higher are identical for *OPT* and *RP*.
(c) All the unmatched L.P. packets in the queues are of index bp_i or higher.

Note that such hypothesis is true for $t_1 = 0$ and $bp_1 = 0$. At time 0, there are no packets in either *OPT* or *RP*'s queue. The requirement for the unmatched L.P. packets is true by default.

We examine the packets admitted to the queue by both *OPT* and *RP* from time t_i on. Assume at time $t'' > t_i$, *OPT* and *RP* disagree with each other for the first time. There are only four general situations:

(1) *OPT* accepted a H.P. packets, while *RP* rejected the packet. This is impossible. The numbers of packets in both queues are the same at time t_i, by condition (a) of the inductive hypothesis. Plus *OPT* and *RP* agree with each other between time (t_i, t''), so the number of packets in both queues are the same just before time t''. If *OPT* accepted a H.P. packet at time t'', that means *OPT*'s queue was not full prior to time t'', neither was *RP*'s queue. *RP* employs greedy strategy for H.P packets. *RP* should have accepted the H.P. packet as well.
(2) *RP* accepted a H.P. packet, while *OPT* rejected. This cannot happen due to Lemma 3.
(3) *OPT* rejected a L.P. packet, while *RP* accepted.
(4) *RP* rejected a L.P. packet, while *OPT* accepted.

In either of the last two cases, we'll find a breaking point $bp_{i+1} > bp_i$ at a time $t_{i+1} > t'' > t_i$. We then show that

$$\frac{T^{OPT}[bp_i, bp_{i+1} - 1]}{T^{RP}[bp_i, bp_{i+1} - 1]} \leqslant C. \qquad \square$$

Let's consider situation (3). At time t'', *OPT* rejected a L.P. packet *sp*, while *RP* accepted. Then by Lemma 5, there is a time $t' > t''$, such that *OPT*'s queue is full at time t' and *OPT* only accepts H.P. packets between time $(t'', t']$. The last packet in *OPT*'s queue at time t' has index $\lceil t' \rceil + B - 1$. We then set $bp_{i+1} = \lceil t' \rceil + B$, and $t_{i+1} = t'$. The reader can verify that $bp_{i+1} > bp_i$ and $t_{i+1} > t_i$.

Lemma 7. *Both OPT's and RP's queues are full at time t'. And neither OPT nor RP was idle during time $(t'', t']$.*

Proof. *OPT* cannot be idle during time $(t'', t']$. Otherwise, we modify *OPT* to accept the additional packet *sp* instead, contradicting the optimality of *OPT*. At time t'', *RP* has one more packet *sp* in its queue compared to that of *OPT*. *RP* is greedy in terms of H.P. packets, so unless *RP*'s queue is full (in which case *RP* has as many packets in its queue as possible), it will always accept a H.P. packet that *OPT* accepts during time $(t'', t']$. *RP* may even accept some L.P. packet during that time, too. So *RP*'s queue will always have equal or more number of packets compared to that of *OPT* until time t'. This implies that *RP*'s queue is full at time t' and never idle during time $(t'', t']$. □

Lemma 8. *At any time we have that $l \times (\alpha - 1) \leqslant f \times \alpha$, where l is the number of unmatched L.P. packets and f is the number of free slots in RP's queue.*

Proof. This is true at time 0. When a L.P. packet is accepted, by definition, the number of unmatched L.P. packet is not more than $\alpha/(\alpha - 1)$ times the total number of the free space, so the bound trivially holds. When a H.P. packet is accepted, the number of free slots decreases by 1, but the number of unmatched L.P. packets is decreased by $\alpha/(\alpha - 1)$, or there is no unmatched L.P. packet. In either case, the claim is true. When a packet departs, the number of free slots increases by 1, the number of unmatched L.P. packets does not increase, hence the bound still holds. □

Corollary 2. *When RP's queue is full, all the L.P. packets are matched in the queue.*

Lemma 9. *At time $t_{i+1} = t'$ and $bp_{i+1} = \lceil t' \rceil + B$, all three conditions in the inductive hypothesis in Theorem* 2 *are satisfied.*

Proof. By Lemma 7, both queues are full, hence condition (a) is satisfied. The packets in *OPT* and *RP*'s queue have indices at most $\lceil t' \rceil + B - 1 < bp_{i+1}$, so condition (b) holds by default. By Corollary 2, condition (c) holds as well. □

We now compare $T^{RP}[bp_i, bp_{i+1} - 1]$ with $T^{OPT}[bp_i, bp_{i+1} - 1]$. In particular, we'll show that

$$T^{OPT}\big[I^{RP}(sp), bp_{i+1} - 1\big] / T^{RP}\big[I^{RP}(sp), bp_{i+1} - 1\big] \leqslant C. \tag{$\star$}$$

Recall that $I^{RP}(sp)$ is the index of packet *sp* on *RP*'s tape. By Lemma 5 and 7, *OPT* has all H.P. packets from $T[I^{RP}(sp)]$ to $T[bp_{i+1} - 1]$, while *RP* has either L.P. or H.P. packets at these positions. We will prove that the total fraction of the L.P. packets at these positions is at most $\alpha/(2\alpha - 1)$, immediately implying ($\star$).

We introduce the following conceptual marking of H.P. packets to L.P. packets for packets from positions $[I^{RP}(sp)]$ to $T[bp_{i+1} - 1]$. Initially, each accepted L.P. packet is unmarked. We mark packets as follows: each accepted H.P. packet with index $I^{RP}(sp)$ or higher would mark up to $\alpha/(\alpha - 1)$ unmarked L.P. packets with indices $I^{RP}(sp)$ or higher. The marking is almost the same as the actual matching used in *RP*, except that only the L.P. packets with indices $I^{RP}(sp)$ or higher are marked. Similar to Lemma 8 and Corollary 2, we conclude the following.

Corollary 3. *When RP's queue is full during any time between $(t'', t']$, all L.P. packets from positions $I^{RP}(sp)$ on are marked in the queue.*

Lemma 10. *At most $\alpha/(2\alpha - 1)$ fraction of the packets between $T^{RP}[I^{RP}(sp)]$ and $T^{RP}[bp_{i+1} - 1]$ are L.P. packets.*

Proof. From now on we completely ignore the packets with indices lower than $I^{RP}(sp)$ in both queues. We use the word "relevant" to refer to packets "with indices $I^{RP}(sp)$ or higher." We consider the longest prefix of consecutive packets that consists of only marked L.P. and H.P. packets in *RP*'s queue. The prefix has to start from either *sp*, or the first packet in *RP*'s queue if *sp* has departed. We use $L(t)$ to denote the length of such longest prefix at any time $t \in [t'', t']$. We prove the following claim: $L(t)$ is at least the number of relevant H.P. packets in *OPT*'s queue at time t.

The base case is at time t'' when *RP* accepted *sp* and *OPT* rejected *sp*. *OPT* has no relevant H.P. packets yet in its queue. For *RP*, initially *sp* is unmarked, so $L(t'') = 0$. The hypothesis is true. Now consider the inductive step. When a H.P. packet *hp* arrives, if *RP* accepts *hp*, then *OPT* accepts *hp* as well by Lemma 3. The number of relevant H.P. packets in *OPT*'s queue increases by 1. Consider the previous valid longest prefix $\mathcal{P}$ in *RP*'s queue, if *hp* immediately follows $\mathcal{P}$, then we append *hp* to $\mathcal{P}$ and the hypothesis is maintained. If *hp* could not extend $\mathcal{P}$, that means there is at least one unmarked L.P. packet *ulp* immediately following $\mathcal{P}$, i.e., all relevant L.P. packets prior to *ulp* in the queue are marked already. By our marking scheme, the arrival of *hp* changes *ulp* to a marked L.P. packet, thus extending $\mathcal{P}$ by at least one, maintaining the hypothesis. If *RP* rejects *hp*, that means *RP*'s queue is full. By Corollary 3, all relevant L.P. packets in *RP*'s queue are marked, so *RP* has the longest possible prefix, maintaining the hypothesis. When a L.P. packet is accepted into *RP*'s queue, the number of relevant H.P. packets in *OPT* doesn't change, neither does the length of the prefix. The hypothesis is also maintained when a packet departs from both queues, by Lemma 7 (neither machine is idle).

The hypothesis implies that *RP* always sends a relevant L.P. packet as a marked L.P. packet, during time $(t'', t']$. Otherwise consider the time t^* just before an unmarked relevant L.P. packet ulp_2 was sent by *RP*, and a H.P. packet was sent by *OPT*. $L(t^*) = 0$, since ulp_2 was the first packet in the queue, but *OPT* has at least one relevant H.P. packet in the queue to be sent, contradicting our claim. At time t', *RP*'s queue is full, and $T^{RP}[bp_{i+1} - 1]$ corresponds to the last packet in *RP*'s queue. By Corollary 3, all the relevant L.P. packets in the queue are marked. We have proved that all L.P. packets from $T[I(sp)]$ to $T[bp_{i+1} - 1]$ are marked. Since each H.P. packet marks at most $\alpha/(\alpha - 1)$ L.P. packets, at most $\alpha/(2\alpha - 1)$ fraction of the these packets are L.P. packets. □

This concludes the discussion about situation (3), now let's consider situation (4). At time t'', *OPT* accepted a L.P. packet *tp*, while *RP* rejected it, so *OPT*'s queue has more packets than that of *RP* at time t''. Then we look at the first time $t' > t''$ when *RP*'s queue contains the same number of packets as that of *OPT*.

Lemma 11. *We can rearrange OPT's queue content during time* $[t'', t']$, *so that the prefix of its queue matches the entire RP's queue content. To be more specific, at any time* $t \in [t'', t']$, *the following conditions hold*:

(1) *OPT always has more packets in its queue than RP for* $t < t'$, *and the same number of packets at time* t'.
(2) *The H.P. packets in OPT's queue are synchronized with that of RP, i.e., if there is a H.P. packet at position* i *in OPT's queue, then there is a H.P. packet at position* i *in RP's queue, and vice versa. In particular, that means the numbers of H.P. packets in both queues are the same.*

Proof. Notice that condition (1) is always true by the definition of t'. We use induction to prove condition (2). The base case is at time t''. By conditions (a) and (b) in Theorem 2, *OPT* only has one extra L.P. packet *tp* at the end of its queue compared to that of *RP*, the hypothesis is true. When a H.P. packet *hp* arrives, by Corollary 1, *OPT* always accepts *hp*. Since *RP* always has no more packets in its queue than *OPT*, *RP* must greedily accept *hp*. Let $P(hp)$ denote the position of *hp* in *RP*'s queue. *OPT* then places *hp* at position $P(hp)$, and places whatever packet was originally at $P(hp)$ at the end of its queue. Such exchange is valid by Lemma 1. Thus the hypothesis is maintained. When a L.P. packet *lp* arrives, the only interesting case is when *RP* accepts *lp* while *OPT* rejects *lp*. Let $P(lp)$ denote the position of *lp* in *RP*'s queue. Since all the H.P. packets are still synchronized, by condition (1), *OPT* must have a L.P. packet at position $P(lp)$ already (otherwise *OPT* has less number of packets, impossible), maintaining the hypothesis. If a packet departs from *RP*, then the same packet must depart from *OPT* as well. If a packet departs from *OPT*, either the same packet departs from *RP*, or *RP* is idle. In any of the departing scenarios, the hypothesis is maintained. □

Corollary 4. *At time* t', *OPT and RP has exactly the same queue contents.*

Proof. By the previous lemma, at time t', we have rearranged *OPT*'s queue so that its prefix matches the entire queue content of *RP*. Since *OPT* has the same number of packets as *RP* at time t', *OPT*'s queue content is exactly that of *RP*. □

During time $[t'', t')$, if *RP* was never idle, then we should ignore the discrepancy at time t''. Here is the reasoning. Since *RP* was never idle, by Lemma 2, *OPT* was never idle either. That means all packets transmitted during time $[t_i, t')$ (with indices bp_i or higher) are the same for *RP* and *OPT*, due to Lemma 11. By Corollary 4, at time t' we can conceptually think *OPT* and *RP* agree with each other during time $[t_i, t']$, even though *OPT* and *RP* made different decisions about accepting the packets. We should then look for the next time t'' where *OPT* and *RP* disagree with each other again.

The interesting case is when *RP* was at least idle once during time $[t'', t')$. The first packet (or the first position, if there is no packet) in *RP* or *OPT*'s queue at time t' has index $\lfloor t' \rfloor$. We then set $bp_{i+1} = \lfloor t' \rfloor$, and $t_{i+1} = t'$. The reader can verify that $t_{i+1} > t_i$ and $bp_{i+1} > bp_i$.

Lemma 12. *At time $t_{i+1} = t'$ and $bp_{i+1} = \lfloor t' \rfloor$, all three conditions in the inductive hypothesis in Theorem* 2 *are satisfied.*

Proof. By the definition of t', condition (a) is satisfied. By Corollary 4, condition (b) is satisfied. The first packet in the queue at time t' has index $bp_{i+1} = \lfloor t' \rfloor$, so condition (c) is satisfied as well. □

We now show that

$$T^{OPT}[bp_i, bp_{i+1} - 1]/T^{RP}[bp_i, bp_{i+1} - 1] \leqslant C. \tag{$*$}$$

Notice that by condition (2) in Lemma 11, the numbers of H.P. packets for *OPT* and *RP* from $T[bp_i]$ to $T[bp_{i+1} - 1]$ are the same. It's sufficient then to argue that the L.P. packets from $T^{RP}[bp_i]$ to $T^{RP}[bp_{i+1} - 1]$ is at least $\alpha/(2\alpha - 1)$ fraction of that of *OPT*. We use the following counting scheme. Every time a L.P. packet is accepted into *RP*'s queue, it contributes another imaginary $(\alpha - 1)/\alpha$ L.P. packet to *RP*'s queue. We allow the imaginary packets to overflow the queue. The imaginary packets can also be "transmitted." *RP* always transmits real packets as in the original schedule, but when it's empty of real packets, we let *RP* transmit a whole imaginary packet if there is any. Once an imaginary packet is transmitted, it's removed from the queue, otherwise it stays in the queue forever. In particular, we have the property that if the L.P. packet is currently in the queue, its corresponding imaginary packet is also in the current queue. This is because *RP* always transmits real L.P. packets first rather than imaginary ones. The imaginary packets also form a separate queue, which is appended to *RP*'s queue of normal packets. We use the word "relevant" here to refer to packets "with indices between bp_i and $bp_{i+1} - 1$."

Lemma 13. *The total number of relevant packets in RP's queue* (*including the imaginary packets generated by the relevant packets*)*, is always no less than the total number of relevant packets in OPT's queue during time* $[t_i, t_{i+1})$.

Proof. The claim is true during time $[t_i, t'')$, since *OPT* and *RP* agree with each other till time t''. Let's consider different scenarios for the inductive step. When a H.P. packet arrives, by the exact argument in Lemma 11, both *OPT* and *RP* accepts. The hypothesis is maintained. When a L.P. packet *lp* arrives, the interesting case is if *OPT* accepts *lp* as a relevant L.P. packet while *RP* rejects. The reason for *RP*'s rejection is that the total number of unmatched L.P. packets in *RP*'s queue is at least $\alpha/(\alpha - 1)$ times the number of free slots. By condition (3) in Theorem 2, all unmatched L.P. packets in *RP*'s queue have indices bp_i or higher. By condition (1) in Lemma 11, they all have indices lower than that of *lp*, which is no more than $bp_{i+1} - 1$. So all the unmatched L.P. packets in *RP*'s queue are relevant and their corresponding imaginary packets must be in *RP*'s queue as well. That means that the number of imaginary L.P. packets in *RP*'s queue currently must be at least the number of free slots. So the total number of packets in *RP*'s queue is B, definitely match that of *OPT*. When a packet departs from *OPT*'s queue, by the inductive hypothesis, a packet, maybe imaginary, must also depart from *RP*'s queue (since *RP* has no less packets than *OPT*). When a packet departs from *RP*'s queue, either a packet must also depart from *OPT*'s queue, or *OPT* is idle (which could happen if the departing packet of *RP* is imaginary). So the hypothesis is maintained for all departing cases. □

From the above lemma we conclude that the total number of relevant L.P. packets *RP* has is at least $\alpha/(2\alpha - 1)$ fraction of that of *OPT*, realizing $(*)$. This concludes the discussion of situation (4) in Theorem 2. We thus obtain the following theorem.

Theorem 3. *RP is* $(2\alpha - 1)/\alpha$*-competitive.*

2.1. Generalization of the nonpreemptive queue model

We now generalize the 2-value model to allow packets take on arbitrary values in $[1, \alpha]$. We prove $\Theta(\log \alpha)$ upper and lower bounds for this model.

2.1.1. Lower bound of $(\log \alpha + 2)/2$

We assume that α is a power of 2. We use packets of value 2^i, for $i = 0, \ldots, \log \alpha$. All the packets arrive during time $(0, 1)$. Packets arrive in increasing order of values, with B packets for each value, i.e., first B packets of value 1 arrives, then if necessary B packets of value 2 arrive, and so on. We let LB denote the inverse of the lower bound, $LB = 2/(\log \alpha + 2)$.

At first B packets of value 1 arrive. To stay $(1/LB)$-competitive, the online algorithm must accept packets with total value at least $LB \times B$. For otherwise the adversary can choose to stop the sequence immediately and accept all B value 1 packets to the queue, while the online algorithm accepted packets with total value less than $LB \times B$. So to stay at least $(1/LB)$-competitive, the online algorithm has to occupy at least LB fraction of the total queue space with value 1 packets. In general, we have the following lemma.

Lemma 14. *After B packets with value 2^i arrive, in order to remain at least $(1/LB)$-competitive, any online algorithm must accept packets with total value at least $LB \times 2^i \times B$. And the longest prefix of the packets in the queue with total value no more than $LB \times 2^i \times B$ occupy at least $(i/2 + 1) \times LB \times B$ slots.*

Proof. We have argued that the base case is true for $i = 0$. Assume this is true for $i = k - 1$, now consider when B packets of value 2^k arrives. To stay $(1/LB)$-competitive, the online algorithm must accept packets with total value at least $LB \times 2^k \times B$, for otherwise the adversary will stop the packet sequence immediately and the optimal offline algorithm will accept all B packets with value 2^k, while the online algorithm accepted packets with total value less than $LB \times 2^k \times B$.

Let P_{k-1} denote the last packet in the longest prefix with total value no more than $LB \times 2^{k-1} \times B$ and N_{k-1} denote the number of packets in the queue after P_{k-1}, before B packets of value 2^k arrived. We need to accept packets so that the total value of packets after P_{k-1} adds up to at least $LB \times 2^{k-1} \times B$. Every one of the N_{k-1} packets has value no more than 2^{k-1}. So the total value of these N_{k-1} packets is no more than $2^k \times \lceil N_{k-1}/2 \rceil$. Consider the first $(LB \times B)/2 - \lceil N_{k-1}/2 \rceil$ value 2^k packets accepted. Now the total value of these value 2^k packets plus the N_{k-1} packets before is no more than $LB \times 2^{k-1} \times B$.

Together with the previous prefix ending with P_{k-1}, we then have a prefix of packets with total value no more than $LB \times 2^k \times B$, and has at least

$$N_{k-1} + \frac{LB \times B}{2} - \left\lceil \frac{N_{k-1}}{2} \right\rceil \geqslant \frac{LB \times B}{2}$$

more packets than the previous one. By the inductive hypothesis, the length of the longest prefix is at least $(k/2 + 1) \times LB \times B$. □

In particular, this lemma applies to $i = \log\alpha$. After B packets with value α arrive, the length of the longest prefix is at least $(\log\alpha/2 + 1) \times LB \times B = B$. That means the total value of packets in the queue is no more than $LB \times 2^{\log\alpha} \times B = \alpha \times LB \times B$. On the other hand, the optimal offline algorithm will accept all B α-valued packets. So the online algorithm cannot be better than $(1/LB)$-competitive.

Theorem 4. *No online algorithm* (*deterministic or randomized*) *has a competitive ratio better than* $(\log\alpha + 2)/2$.

2.1.2. Upper bound of $2\log\alpha + 2$

We now present a deterministic algorithm, which is factor 4 away from our lower bound. First we discretize the packet values. We restrict ourselves to instances where packet values are of powers of 2. Given any unrestricted instance $\mathcal{I}$, we round down the packet values to the nearest powers of 2 to create a restricted instance $\mathcal{I}'$.

Lemma 15. *The optimal offline value for* $\mathcal{I}'$ *is at least half of that of* $\mathcal{I}$.

Proof. Just consider the packets accepted by the optimal offline algorithm for $\mathcal{I}$. The same set of packets is a valid solution for $\mathcal{I}'$. For each accepted packet, the rounded down value is at least half the original value. So we have a solution for $\mathcal{I}'$, whose total value is at least half the optimal solution for $\mathcal{I}$. □

From now on we only consider the restricted instances. We first introduce a Round Robin method, and then adapt the method into a valid online algorithm.

We divide the queue into $\lfloor\log\alpha\rfloor + 1$ equal portions, with the ith portion dedicated to packets of value 2^i, for $i = 0, 1, \ldots, \lfloor\log\alpha\rfloor$. Upon the arrival of a value 2^i packet, the Round Robin method greedily accepts if the ith portion has more than one free slot. At each integral time unit, we transmit $1/l$ fractional packet each from the current l non-empty portions. In the case of some portions has less than $1/l$ fractional packet, we divide the extra transmission power evenly among the rest of the non-empty portions, guaranteeing that we transmit one packet per time unit, unless all portions are empty. From [1], we have the following lemma (Corollary 3 in [1]).

Lemma 16. *The total value of the value* 2^i *packets accepted by the Round Robin method is at least* $1/(\lfloor\log\alpha\rfloor + 1)$ *fraction of that of the optimal offline algorithm.*

We can simulate the Round Robin method as follows. We run the Round Robin method in the background, and make acceptance/rejection decisions based on the Round Robin method. Our online algorithm accepts a packet if and only if it's accepted in the Round Robin method. Since the Round Robin method transmits one packet per unit time, the total number of packets in the online algorithm's queue equals to that of the Round Robin method. So we can indeed accept every packet the Round Robin method accepts.

Theorem 5. *Using the Round Robin method as a guide to decide which packet to accept, the online algorithm is* $(2\log\alpha + 2)$*-competitive.*

Proof. Since the online algorithm accepts exactly the same packets as the Round Robin method, the online algorithm has the same competitive ratio as the Round Robin method. The Round Robin method is $(\log\alpha + 1)$-competitive with respect to the restricted instance by Lemma 16, hence $(2\log\alpha + 2)$-competitive with respect to the original instance by Lemma 15. $\square$

3. Preemptive queueing policy

Preemptive queueing policies give online algorithms the ability to preempt (drop) a packet accepted to the queue before its transmission, hence better performance ratios can be derived. To maintain the FIFO order, packets are always added to the end of the queue, and transmitted from the beginning of the queue. In addition, a packet can be dropped from any position in the current queue. The rest of the model setup is identical to that of the non-preemptive model. We concentrate on the model where packets can take on values in the range $[1, \alpha]$. We first consider the case $B = 2$. We show the $(5 + \sqrt{13})/6 \approx 1.434$ upper and lower bounds in terms of competitive ratio for deterministic algorithms.

3.1. Lower bound construction

We start with some intuition via simple constructions that give loose bounds. From now on, when the context is clear, we may refer to a packet by its value; for instance, "packet p" or "p" refers to the packet with value p. We use the following notations: $p \equiv q$ means p and q are the same packet; $p = q$ means packet p and q has the same value. All inequalities refer to the values of packets only.

The basic intuition behind the lower bound is as follows: Suppose the online algorithm currently has two packets in the queue, with values p and q respectively. Suppose $p < q$ and p has to be transmitted before q. Without knowing future arrivals, the online algorithm faces a dilemma: either to transmit p and let q remain in the queue, which puts the online algorithm in danger of discarding q later; or discard p and transmit q, which puts the online algorithm in danger of being idle the next time unit. Thus the adversary, which determines the "difficult" packet sequences, will try to balance these two situations in order to maximize the lower bound.

We introduce the following notation for the packet arrival sequence: packets (values) are listed in their arrival order, with packets arrived within the same time unit grouped into parentheses. For instance, $(a, b)(c, d)$ means packets valued a and b arrived during time

$(0,1)$, with a ahead of b. And packets valued c and d arrived during time $(1,2)$, with c ahead of d.

Consider the following initial sequence: $(1,\alpha)$ with $\alpha > 1$ a parameter to be determined later. If the online algorithm transmits α at time 1, then the sequence ends. *OPT* transmits both 1 and α in two time units. The competitive ratio is then $(1+\alpha)/\alpha$. Otherwise the online algorithm can transmit 1 and put α in the queue at time 1, then the whole sequence becomes $(1,\alpha)(\alpha,\alpha)$. *OPT* transmits all three α-valued packets, while the online algorithm has to discard one such packet during the second time unit. So the competitive ratio becomes $3\alpha/(2\alpha+1)$. Balancing the two ratios $(1+\alpha)/\alpha = 3\alpha/(2\alpha+1)$, we have $\alpha = (3+\sqrt{13})/2$, and the lower bound is approximately 1.303. This lower bound construction takes advantage of the online algorithm's inability to distinguish between two sequences $(1,\alpha)$ and $(1,\alpha)(\alpha,\alpha)$.

Now we try to improve the bound via extending the whole sequences to $(1,\alpha_1)$, or $(1,\alpha_1)(\alpha_1,\alpha_2)$, or $(1,\alpha_1)(\alpha_1,\alpha_2)(\alpha_2,\alpha_2)$. In the future, we'll just write one single sequence $(1,\alpha_1)(\alpha_1,\alpha_2)(\alpha_2,\alpha_2)$, with the understanding that each prefix serves as a possible sequence of packet arrivals. If at time 1 the online algorithm transmits α_1, then the sequence becomes $(1,\alpha_1)$. *OPT* would have transmitted packet 1 at time 1 and packet α_1 at time 2, so the ratio is $(1+\alpha_1)/\alpha_1$. Otherwise, the online algorithm can at best transmit packet 1 at time 1. Now between time $(1,2)$, α_1 and α_2 arrive. Let's consider time 2. If the online algorithm transmits α_2, then the sequence becomes $(1,\alpha_1)(\alpha_1,\alpha_2)$. *OPT* would have transmitted the first α_1 at time 1, the second α_1 at time 2, and α_2 at time 3, for a total of $\alpha_1+\alpha_1+\alpha_2$. The online algorithm transmits packet 1 at time 1 and packet α_2 at time 2, so the ratio is $(\alpha_1+\alpha_1+\alpha_2)/(1+\alpha_2)$. Otherwise, the online algorithm can at best transmit α_1 at time 2. The sequence then becomes $(1,\alpha_1)(\alpha_1,\alpha_2)(\alpha_2,\alpha_2)$. *OPT* would have transmitted α_1 at time 1, the first α_2 at time 2, the second α_2 at time 3, and the third α_2 at time 4, for a total of $\alpha_1+\alpha_2+\alpha_2+\alpha_2$. The online algorithm transmits 1 at time 1, α_1 at time 2, and α_2's at times 3 and 4, so the ratio is $(\alpha_1+\alpha_2+\alpha_2+\alpha_2)/(1+\alpha_1+\alpha_2+\alpha_2)$. Setting the three ratios equal, we have $\alpha_1 = (3+\sqrt{5})/2$, $\alpha_2 = (9+4\sqrt{5})/2$, and the lower bound is approximately 1.382.

Thus if we continue the trend, we can improve the bound further. Consider the following general sequence $(\alpha_0 = 1,\alpha_1)(\alpha_1,\alpha_2)\dots(\alpha_{k-1},\alpha_k)(\alpha_k,\alpha_k)$. We use L_k to denote the lower bound derived from the sequence ending with α_k. We have the following equations, derived from various scenarios similar to the two previous constructions:

$$\begin{aligned}L_k &= \frac{\alpha_0+\alpha_1}{\alpha_1} = \frac{\alpha_1+\alpha_1+\alpha_2}{\alpha_0+\alpha_2} = \frac{\alpha_1+\alpha_2+\alpha_2+\alpha_3}{\alpha_0+\alpha_1+\alpha_3} = \cdots\\ &= \frac{\alpha_1+\alpha_2+\cdots+\alpha_{k-1}+\alpha_{k-1}+\alpha_k}{\alpha_0+\alpha_1+\cdots+\alpha_{k-2}+\alpha_k}\\ &= \frac{\alpha_1+\alpha_2+\cdots+\alpha_{k-1}+\alpha_k+\alpha_k+\alpha_k}{\alpha_0+\alpha_1+\cdots+\alpha_{k-2}+\alpha_{k-1}+\alpha_k+\alpha_k}.\end{aligned}$$

Solving the above equations, we have

$$\alpha_i = \frac{1}{3}\left(\frac{\sqrt{13}+5}{2}\right)^i + \frac{2}{3}\left(\frac{\sqrt{13}-1}{2}\right)^i$$

and as $k \to \infty$, $L_k \to (\sqrt{13} + 5)/6 \approx 1.434$. Appendix A provides the details of the calculation. From now on let APX denote the constant $(\sqrt{13} + 5)/6$.

Theorem 6. *No deterministic online algorithm can achieve a competitive ratio better than $APX - \varepsilon$, for any constant $\varepsilon > 0$.*

Proof. For any $\varepsilon > 0$, we can find a particular value j, such that $L_j > APX - \varepsilon$. By applying the sequence $(1, \alpha_1)(\alpha_1, \alpha_2) \ldots (\alpha_{j-1}, \alpha_j)(\alpha_j, \alpha_j)$ and the analysis above, we can conclude that any deterministic algorithm has a competitive ratio no better than $APX - \varepsilon$. $\square$

3.2. *The Ratio algorithm*

The α_i series provides much insight on designing an online algorithm with the matching upper bound. We see that the α_i series is a combination of two geometric series of powers $(\sqrt{13} + 5)/2$ and $(\sqrt{13} - 1)/2$, respectively. The dominating factor is $(\sqrt{13} + 5)/2$, since $\alpha_i/(\alpha_{i-1})$ increases and converges to this value as $i \to \infty$. Let r denote the constant $(\sqrt{13} + 5)/2$. For any k, consider the packet arrival sequence $(1, r)(r, r^2) \ldots (r^{k-1}, r^k)(r^k, r^k)$, we have the following observations:

$$\begin{aligned} \frac{r + r^2 + \cdots + r^{k-1} + r^{k-1} + r^k}{1 + r + \cdots + r^{k-2} + r^k} &= \frac{\sum_{i=1}^{k-1} r^i + r^{k-1} + r^k}{\sum_{i=1}^{k-2} \sum r^i + r^k} \\ &< \frac{r^{k-1} \cdot \frac{r}{r-1} + r^{k-1} + r^k}{r^{k-2} \cdot \frac{r}{r-1} + r^k} = APX, \end{aligned} \tag{1}$$

$$\frac{r + r^2 + \cdots + r^{k-1} + r^k + r^k + r^k}{1 + r + \cdots + r^{k-1} + r^k + r^k} < \frac{r^k \cdot \frac{r}{r-1} + r^k + r^k}{r^k \cdot \frac{r}{r-1} + r^k} = APX. \tag{2}$$

Further, as $k \to \infty$, (1) and (2) approach equality. So in some sense, the geometric series $1, r, r^2, \ldots$ alone can almost provide the lower bound on the competitive ratio, except for small k's. The other factor $(\sqrt{13} - 1)/2$ is only used by the adversary to make the lower bound high for small k's as well. The online algorithm, however, only has to guarantee the competitive ratio stays below (not necessarily equal) to APX for all k. Thus the online algorithm has relatively less work to do.

We will discuss about OPT's behavior first, motivating the design of the online algorithm later. We use A_j and B_j to denote the two most valuable packets arrived during time $(j - 1, j)$, with A_j arriving ahead of B_j. We call a packet being "buffered" by an algorithm if it arrived during time $(j - 1, j)$, and still remains in the queue after time j. We use TR_j to denote the packet transmitted at time j, BF_j the packet buffered at time j, MV_j the most valuable packet arrived after TR_j during time $(j - 1, j)$. Superscripts are used to denote the algorithms and omitted when the context is clear. For instance, TR_j^{OPT} is the packet transmitted by OPT at time j. We also introduce the following notation: if an algorithm transmits packet p and buffered packet q at time j, $p[q]$ denotes such an action ([0] or [ ] means no packet is buffered).

Lemma 17. *Without loss of generality, we assume that if a packet is not to be transmitted by OPT, OPT will not accept that packet into the queue. Similarly, if OPT buffered a packet at time j, we would assume OPT transmits that packet at time $j+1$, i.e., $BF_j > 0$ implies $BF_j \equiv TR_{j+1}$.*

Lemma 18. *Without loss of generality, either $TR_j \equiv BF_{j-1} \equiv MV_{j-1} > \min(A_j, B_j)$, or $TR_j \equiv A_j$, or $TR_j \equiv B_j$.*

Proof. It's clear that BF_{j-1} has to arrive during time $(j-2, j-1)$, and has to arrive after TR_{j-1} by the FIFO ordering. The most valuable packet satisfying these criterions is the only choice for BF_{j-1}, so $BF_{j-1} \equiv MV_{j-1}$.

By the previous lemma, if $BF_{j-1} > 0$, then it must be transmitted at time j. If $MV_{j-1} \leqslant \min(A_j, B_j)$, then consider the following modification to *OPT*. If *OPT* executed $MV_{j-1}[r]$ at time j for some packet $r > 0$, then modify *OPT* to execute $A_j[B_j]$ instead. The packets transmitted by *OPT* at time j and $j+1$ originally were MV_{j-1} and r, vs. A_j and B_j after the modification. It is clear that $MV_{j-1} + r \leqslant A_j + B_j$, since $r \leqslant \max(A_j, B_j)$. Thus doing so would not violate the optimality of *OPT*. If *OPT* executed $MV_{j-1}[\ ]$ at time j, then modify *OPT* to execute $\max(A_j, B_j)[\ ]$ instead. Clearly, $\max(A_j, B_j) \geqslant MV_{j-1}$, maintaining the optimality of *OPT*.

Otherwise, $BF_{j-1} = 0$. The packet transmitted at time j must arrive during time $(j-1, j)$. Assume that *OPT* transmitted a third packet p at time j, where $p < \min(A_j, B_j)$. If *OPT* executed $p[\ ]$ at time j, then we modify *OPT* to execute $\max(A_j, B_j)[\ ]$ instead, increasing the optimal value. Clearly, this cannot happen. If *OPT* executed $p[r > 0]$ at time j, then we modify *OPT* to execute $A_j[B_j]$ instead. Again $p + r < A_j + B_j$, a contradiction to the optimality of *OPT*. □

From the previous lemma, it's clear that for *OPT*'s decision to transmit and buffer packets at time j, the only packets worth considering are the two most valuable packets out of the following packets: MV_{j-1}, A_j, and B_j. We now introduce the Ratio algorithm (denoted by *RA*). *RA* will also keep track of the three packets $MV_{j-1}^{RA} \equiv BF_{j-1}^{RA}$, A_j, and B_j. Notice that *RA* will always buffer the packet MV_{j-1}^{RA}, regardless whether or not it will be transmitted at time j. Of these three packets, the two most valuable packets F_j and S_j (with F_j arriving ahead of S_j) will be considered and possibly put in the queue. If $S_j \geqslant F_j \times r$, then F_j will be dropped and $S_j \equiv TR_j$, the most valuable packet arriving after S_j will become BF_j. We call F_j being "preempted" by S_j. Otherwise $S_j < F_j \times r$, then $F_j \equiv TR_j$ and $S_j \equiv BF_j$. Figure 1 shows the pseudo-code for the Ratio algorithm. We use B_1^t to denote the first packet in the queue at time t, and B_2^t the second packet. Recall that B_1^t and B_2^t also refers to the values of these packets, $B_i^t = 0$ means there is no packet in the ith slot. Obviously, $B_1^t = 0$ implies $B_2^t = 0$.

Theorem 7. *The Ratio algorithm is $((5+\sqrt{13})/6)$-competitive.*

The full proof is a detailed case analysis. We first prove the following simple case.

Lemma 19. *If at each transmission time j, $F_j \geqslant S_j$, then RA is optimal.*

```
RATIO
1  For the arrival of a new packet $p$ at time $t$
2  IF $B_1^t = 0$
3      ACCEPT $p$ into the queue, so now $B_1^t = p$
4  ELSE
5      IF $p \leqslant B_1^t$ and $p \leqslant B_2^t$
6          DROP $p$
7      ELSE
8          DROP the smaller of $B_1^t$ and $B_2^t$, so now $B_2^t = 0$
9          ACCEPT $p$ into the queue, so now $B_2^t = p$
10         IF $B_2^t / B_1^t \geqslant r$
11             DROP $B_1^t$, so now $B_1^t = p$, and $B_2^t = 0$
12 At transmission time $j$
13 IF queue is not empty
14     Transmit the first packet in the queue, so now $B_2^j = 0$
```

Fig. 1. Pseudo-code for the Ratio algorithm.

Proof. Since $F_j \geqslant S_j$, by the definition of *RA*, there are two possibilities for the orderings of BF_{j-1}^{RA}, A_j, and B_j below.

(1) $BF_{j-1}^{RA} > \max(A_j, B_j)$, in which case $F_j \equiv BF_{j-1}^{RA}$ and $S_j \equiv \max(A_j, B_j)$.
(2) $A_j \geqslant B_j \geqslant BF_{j-1}^{RA}$, in which case $F_j \equiv A_j$ and $S_j \equiv B_j$.

We are going to prove $TR_j^{RA} \equiv TR_j^{OPT}$ and $BF_j^{RA} \equiv MV_j^{OPT}$, by induction on j. Clearly, when $j = 0$, the hypothesis is true by default. We now look at time j, assuming the hypothesis is true up to time $j-1$. In case (1) above, $MV_{j-1}^{OPT} \equiv BF_{j-1}^{RA} > \max(A_j, B_j) \geqslant \min(A_j, B_j)$. By Lemma 18,

$$TR_j^{OPT} \equiv BF_{j-1}^{OPT} \equiv MV_{j-1}^{OPT}.$$

By the definition of *RA*, $TR_j^{RA} \equiv BF_{j-1}^{RA} \equiv TR_j^{OPT}$, implying

$$BF_j^{RA} \equiv MV_j^{RA} \equiv MV_j^{OPT}.$$

In case (2), $BF_{j-1}^{RA} \equiv MV_{j-1}^{OPT} \leqslant \min(A_j, B_j)$. By Lemma 18, *OPT* must transmit one of A_j and B_j. If $TR_j^{OPT} \equiv B_j$, we modify *OPT* so that $TR_j^{OPT} \equiv A_j$. It's clear such modification wouldn't hurt *OPT* because $A_j \geqslant B_j$. By the definition of *RA*, $TR_j^{RA} \equiv A_j \equiv TR_j^{OPT}$, implying

$$B_j \equiv BF_j^{RA} \equiv MV_j^{RA} \equiv MV_j^{OPT}.$$

Thus *RA* transmits the same packets as *OPT* at all times. □

Lemma 20. *Consider the first time j where OPT and RA differ from each other, i.e., $TR_j^{OPT} \not\equiv TR_j^{RA}$, then either $TR_j^{OPT} \equiv F_j$ and $TR_j^{RA} \equiv S_j$, or $TR_j^{OPT} \equiv S_j$ and $TR_j^{RA} \equiv F_j$.*

Proof. Since $TR^{OPT}_{j-1} \equiv TR^{RA}_{j-1}$, so $MV^{OPT}_{j-1} \equiv BF^{RA}_{j-1}$. Thus F_j and S_j are also the two most valuable packets to consider for *OPT* at time j, by Lemma 18, $TR^{OPT}_j \equiv F_j$ or $TR^{OPT}_j \equiv S_j$. By the definition of *RA*, either $TR^{RA}_j \equiv F_j$, or $TR^{RA}_j \equiv S_j$. $\square$

The proof of the following lemma is trivial by the definition of *RA*.

Lemma 21. *Either* $TR^{RA}_j = \max(A_j, B_j)$*, or* $BF^{RA}_j = \max(A_j, B_j)$*, for all* j*.*

We use $OPT[i, j]$ ($RA[i, j]$, respectively) to denote the total values of packets transmitted by *OPT* (*RA*, respectively) during time $[i, j]$. As in the nonpreemptive situation, we start from the first time j_0 where *OPT* and *RA* transmits different packets, and we'll find a future time j', where $OPT[j_0, j']/RA[j_0, j'] \leqslant APX$, etc. For simplicity, assume $j_0 = 0$. From Lemmas 19 and 20, we know $F_0 < S_0$ and there are two cases.

The first (simple) case is $TR^{RA}_0 \equiv S_0$ and $TR^{OPT}_0 \equiv F_0$. Then $BF^{OPT}_0 \equiv TR^{OPT}_1 \equiv S_0$, since $F_0 < S_0$. Thus

$$\frac{OPT[0,1]}{RA[0]} = \frac{F_0 + S_0}{S_0} \leqslant \frac{1+r}{r} \leqslant APX.$$

Notice that we didn't count TR^{RA}_1 in the comparison. If $BF^{OPT}_1 = 0$, then we can totally ignore TR^{RA}_1, and start from after the transmission at time 1 as the beginning of time. This is because *OPT* will only have packets arrived after time 1 to consider after time 1, which are all available to *RA* as well. *RA* might have buffered a packet at time 1, but this only gives *RA* more choices on packets, Lemmas 19 and 20 should still apply from time 1 onwards. Else $BF^{OPT}_1 > 0$, since $TR^{OPT}_1 = S_0$, BF^{OPT}_1 must be $\max(A_1, B_1)$. By Lemma 21, we have the following two situations:

- $BF^{OPT}_1 \equiv BF^{RA}_1$. This is the ideal case, Lemmas 19 and 20 should be applicable from time 1 on since *OPT* and *RA* buffered the same packet. We thus look for the next time *OPT* and *RA* disagree on transmission, and so on.
- $BF^{OPT}_1 \equiv TR^{RA}_1$. Let $k > 1$ be the first such time that *OPT* didn't buffer a packet transmitted by *RA* at time k, i.e., during time $j \in [1, k)$, $TR^{RA}_j \equiv BF^{OPT}_j$. Or let $k > 1$ be the first such time that $TR^{RA}_k = 0$, whichever appears first. Then
$$\frac{OPT[2,k]}{RA[1,k]} = 1,$$
since $TR^{OPT}_l \equiv BF^{OPT}_{l-1} \equiv TR^{RA}_{l-1}$, for $l \in [2, k]$. We then want to claim that from time k onward, *OPT* and *RA* are synchronized again. If $TR^{RA}_k = 0$, then it has to be the case that no packet arrived during time $(k-1, k)$. Thus $BF^{OPT}_k = BF^{RA}_k = 0$, so Lemmas 19 and 20 apply from time k onwards. Otherwise, consider time $k-1$, we know $BF^{OPT}_{k-1} \equiv TR^{RA}_{k-1} \neq 0$, so either $BF^{OPT}_k = 0$, or $BF^{OPT}_k \equiv \max(A_k, B_k)$. If $BF^{OPT}_k = 0$, then by the same argument before, Lemmas 19 and 20 applies from time k again. If $BF^{OPT}_k \equiv \max(A_k, B_k) \not\equiv TR^{RA}_k$, then it has to be that $BF^{RA}_k \equiv \max(A_k, B_k)$ due to Lemma 21. So Lemmas 19 and 20 again applies from time k onwards.

This completes the argument for the first case where $TR^{RA}_0 = S_0$ and $TR^{OPT}_0 = F_0$. We concluded that there exists a time k, such that $OPT[0, k]/RA[0, k] \leqslant APX$, plus *OPT* and

RA are synchronized once more at time k, enabling Lemmas 19 and 20 again. Notice that the synchronize step is needed for a particular time i, where $TR_{i-1}^{RA} \equiv TR_i^{OPT}$. The analysis above is for $i = 1$, but the same analysis applies for all such time i. In the remaining analysis, we won't explicitly bring up this situation, with the understanding that it can be dealt with.

The second case is more involved, where $TR_0^{RA} \equiv F_0$ and $TR_0^{OPT} \equiv S_0$ (thus $BF_0^{RA} = S_0$). We keep the inequalities (1) and (2) in mind, these are the worst case situations for *RA*, where $S_j = r \times F_j$. We shall transform all other situations to this standard case, via the following two mathematical lemmas.

Lemma 22. *Let Y, Z, U_i, V_i $(i = 1, \ldots, n)$, W_j, X_j $(j = 1, \ldots, m)$ be positive real values, for some n and m. If*

$$\frac{Y}{Z} = \frac{U_1 + U_2 + \cdots + U_n - W_1 - W_2 - \cdots - W_m}{V_1 + V_2 + \cdots + V_n - X_1 - X_2 - \cdots - X_m} \quad \text{and} \quad \max_i \frac{U_i}{V_i} \leqslant \min_j \frac{W_j}{X_j},$$

then $Y/Z \leqslant \max_i\{U_i/V_i\} \leqslant \min_j\{W_j/X_j\}$.

Proof. We repeatedly use the following fact about 4 positive real values A, B, C, and D that satisfy

$$\frac{A}{B} \leqslant \frac{C}{D} : \frac{A-C}{B-D} \leqslant \frac{A}{B} \leqslant \frac{A+C}{B+D} \leqslant \frac{C}{D}. \qquad \square$$

Lemma 23. *The following is true for all $M > 0$:*

$$\frac{\frac{M \times r}{r-1} + 2M}{\frac{M}{r-1} + 2M} = APX \quad \text{and} \quad \frac{\frac{M \times r}{r-1} + M + M \times r}{\frac{M}{r-1} + M \times r} = APX.$$

Proof.

$$\frac{\frac{M \times r}{r-1} + 2M}{\frac{M}{r-1} + 2M} = \frac{\frac{r}{r-1} + 2}{\frac{1}{r-1} + 2} = \frac{3r-2}{2r-1} = APX,$$

$$\frac{\frac{M \times r}{r-1} + M + M \times r}{\frac{M}{r-1} + M \times r} = \frac{\frac{r}{r-1} + 1 + r}{\frac{1}{r-1} + r} = \frac{r^2 + r - 1}{r^2 - r + 1} = APX. \qquad \square$$

Lemma 24. *We claim the following inductive hypothesis for any time $j > 0$:*

- *Either $OPT[0, j]/RA[0, j] \leqslant APX$, or $OPT[0, j+1]/RA[0, j] \leqslant APX$, or $OPT[0, j+2]/RA[0, j+1] \leqslant APX$, or $OPT[0, j+1]/RA[0, j+1] \leqslant APX$. And there is an appropriate time j' such that OPT and RA are synchronized and Lemmas* 19 *and* 20 *apply.*
- *Or $BF_j^{OPT} = 0$ and*

$$\frac{OPT[0, j]}{RA[0, j]} = \frac{\frac{BF_j^{RA} \times r}{r-1} + U_i - W_j}{\frac{BF_j^{RA}}{r-1} + V_i - X_j},$$

where $U_i/V_i \leqslant APX \leqslant W_j/X_j$. Thus by Lemma 22, we can ignore the U_i, V_i, W_j, X_j terms while trying to prove the ratio APX, and only conceptually think

$$\frac{OPT[0,j]}{RA[0,j]} = \frac{\frac{BF_j \times r}{r-1}}{\frac{BF_j}{r-1}}.$$

Proof. The base case is for $j = 0$: if $BF_0^{OPT} \neq 0$, then $TR_1^{OPT} \equiv BF_0^{OPT} \leqslant F_0$ by the definition of F_0 and S_0. Then either $TR_1^{RA} = S_0$, in which case

$$\frac{OPT[0,1]}{RA[0,1]} \leqslant \frac{S_0 + F_0}{F_0 + S_0} \leqslant 1.$$

And after time 1 OPT and RA are synchronized again since $MV_1^{OPT} = BF_1^{RA} = \max(A_1, B_1)$. Or $TR_1^{RA} \not\equiv S_0$, that means S_0 was thrown away. By the definition of RA, it must be that $TR_1^{RA} \geqslant r \times S_0$. Then $TR_2^{OPT} \equiv TR_1^{RA}$, since $TR_1^{OPT} \equiv BF_0^{OPT} \leqslant F_0 < S_0$. Thus

$$\frac{OPT[0,2]}{RA[0,1]} = \frac{S_0 + TR_1^{OPT} + TR_2^{OPT}}{F_0 + TR_1^{RA}} < \frac{S_0 + F_0 + TR_1^{RA}}{F_0 + S_0 + TR_1^{RA} \cdot \frac{r-1}{r}} < \frac{r}{r-1} \leqslant APX.$$

The other case is $BF_0^{OPT} = 0$. Also we know $F_0 > S_0/r$ and $BF_0^{RA} = S_0$ from the definition of RA:

$$\frac{OPT[0,0]}{RA[0,0]} = \frac{S_0}{F_0} = \frac{\frac{S_0 \times r}{r-1} - \frac{S_0}{r-1} + 0}{\frac{S_0}{r-1} - \frac{S_0}{r(r-1)} + \left(F_0 - \frac{S_0}{r}\right)},$$

where

$$\frac{\frac{S_0}{r-1}}{\frac{S_0}{r(r-1)}} = r > APX > \frac{0}{F_0 - \frac{S_0}{r}},$$

satisfying the inductive hypothesis. This completes the base case analysis.

Assume at time $l-1$ the inductive hypothesis is true. If at time $l-1$ the first condition of the inductive hypothesis holds, then we skip to the appropriate time where OPT and RA are synchronized once more and apply Lemmas 19 and 20. Otherwise if there is no new packet arriving during time $(l-1, l)$, then $TR_j^{RA} \equiv BF_{l-1}^{RA}$ while $TR_j^{OPT} = 0$, since $BF_{j-1}^{OPT} = 0$. Then

$$\frac{OPT[0,l]}{RA[0,l]} = \frac{\frac{BF_{l-1}^{RA} \times r}{r-1}}{\frac{BF_{l-1}^{RA}}{r-1} + BF_{l-1}^{RA}} \leqslant APX$$

and OPT and RA are synchronized again after time l.

So the only case left is that the second condition of the inductive hypothesis held at time $l-1$ and at least one new packet arrived during time $(l-1, l)$, and so both $F_l \neq 0$ and $S_l \neq 0$. Here are the possibilities:

(1) If $TR_l^{RA} \equiv S_l$. Then $S_l/F_l \geqslant r$, and $F_l \geqslant BF_{l-1}^{RA}$. Then either $TR_l^{OPT} \equiv S_l$, or $TR_l^{OPT} \leqslant F_l$ and $TR_{l+1}^{OPT} \equiv S_l$. Thus $OPT[0,l]/RA[0,l]$ or $OPT[0,l+1]/RA[0,l]$ is no more than:

$$\frac{\frac{BF_{l-1}^{RA}\times r}{r-1} + F_l + S_l}{\frac{BF_{l-1}^{RA}}{r-1} + S_l} = \frac{\frac{F_l\times r}{r-1} + F_l + F_l \times r + (S_l - F_l \times r) - \frac{(F_l - BF_{l-1}^{RA})\times r}{r-1}}{\frac{F_l}{r-1} + F_l \times r + (S_l - F_l \times r) - \frac{F_l - BF_{l-1}^{RA}}{r-1}}$$
$$\leqslant APX.$$

The last inequality is due to Lemmas 22 and 23.

(2) Else then $TR_l^{RA} \equiv F_l$ and $BF_l^{RA} \equiv S_l$. If $BF_l^{OPT} \neq 0$. We consider then the following two subcases.
The first subcase is $TR_{l+1}^{RA} \equiv BF_l^{RA} \equiv S_l$. Then we will show below that $OPT[0,l+1]/RA[0,l+1] \leqslant APX$, via another case analysis.
If $F_l, S_l \geqslant BF_{l-1}^{RA}$, we know OPT transmits no more than a combined $F_l + S_l$ at times l and $l+1$. Then

$$\begin{aligned}
\frac{OPT[0,l+1]}{RA[0,l+1]} &\leqslant \frac{\frac{BF_{l-1}^{RA}\times r}{r-1} + F_l + S_l}{\frac{BF_{l-1}^{RA}}{r-1} + F_l + S_l}\\
&= \frac{\frac{BF_{l-1}^{RA}\times r}{r-1} + 2BF_{l-1}^{RA} + (F_l + S_l - 2BF_{l-1}^{RA})}{\frac{BF_{l-1}^{RA}}{r-1} + 2BF_{l-1}^{RA} + (F_l + S_l - 2BF_{l-1}^{RA})}\\
&\leqslant \frac{3r-2}{2r-1}\\
&= APX.
\end{aligned}$$

Else then $F_l = BF_{l-1}^{RA} > S_l$,we know OPT transmits no more than a combined $2S_l$ at times l and $l+1$ since $BF_{l-1}^{OPT} = 0$. So

$$\begin{aligned}
\frac{OPT[0,l+1]}{RA[0,l+1]} &\leqslant \frac{\frac{BF_{l-1}^{RA}\times r}{r-1} + 2S_l}{\frac{BF_{l-1}^{RA}}{r-1} + BF_{l-1}^{RA} + S_l} = \frac{\frac{BF_{l-1}^{RA}\times r}{r-1} + 2BF_{l-1}^{RA} - (2BF_{l-1}^{RA} - 2S_l)}{\frac{BF_{l-1}^{RA}}{r-1} + 2BF_{l-1}^{RA} - (BF_{l-1}^{RA} - S_l)}\\
&\leqslant APX.
\end{aligned}$$

The second subcase is $TR_{l+1}^{RA} > BF_l^{RA} \equiv S_l$. Then it has to be that $TR_{l+1}^{RA} \geqslant S_l \times r$, and $TR_{l+2}^{OPT} \equiv TR_{l+1}^{RA}$ since $TR_{l+1}^{OPT} \equiv BF_l^{OPT}$. Compare $OPT[0,l+2]/RA[0,l+1]$ with the two calculations of $OPT[0,l+1]/RA[0,l+1]$ above, OPT added TR_{l+1}^{RA} to the numerator, while RA added at least $TR_{l+1}^{RA} - S_l$ to the denominator. And $TR_{l+1}^{RA}/(TR_{l+1}^{RA} - S_l) \leqslant r/(r-1) < APX$. This proves $OPT[0,l+2]/RA[0,l+1] \leqslant APX$.

(3) Else then $TR_l^{RA} \equiv F_l$, $BF_l^{RA} \equiv S_l$, and $BF_l^{OPT} = 0$.

If $F_l \geqslant S_l \geqslant BF_{l-1}$, then $TR_l^{OPT} \equiv F_l$.

$$\frac{OPT[0,l]}{RA[0,l]} = \frac{\frac{BF_{l-1}^{RA}\times r}{r-1} + F_l}{\frac{BF_{l-1}^{RA}}{r-1} + F_l} = \frac{\frac{S_l\times r}{r-1} + F_l - \frac{(S_l - BF_{l-1}^{RA})\times r}{r-1}}{\frac{S_l}{r-1} + F_l - \frac{S_l - BF_{l-1}^{RA}}{r-1}}.$$

Else if $BF_{l-1} \leqslant F_l < S_l < F_l \times r$, then $TR_l^{OPT} \equiv S_l$.

$$\frac{OPT[0,l]}{RA[0,l]} = \frac{\frac{BF_{l-1}^{RA}\times r}{r-1} + S_l}{\frac{BF_{l-1}^{RA}}{r-1} + F_l} = \frac{\frac{F_l\times r}{r-1} + S_l - \frac{(F_l - BF_{l-1}^{RA})\times r}{r-1}}{\frac{F_l}{r-1} + F_l - \frac{F_l - BF_{l-1}^{RA}}{r-1}}$$

$$= \frac{\frac{F_l\times r}{r-1} + S_l - \frac{(F_l - BF_{l-1}^{RA})\times r}{r-1}}{\frac{F_l\times r}{r-1} - \frac{F_l - BF_{l-1}^{RA}}{r-1}} = \frac{\frac{S_l\times r}{r-1} - \frac{(F_l - BF_{l-1}^{RA})\times r}{r-1} + \frac{F_l\times r - S_l}{r-1}}{\frac{S_l}{r-1} - \frac{F_l - BF_{l-1}^{RA}}{r-1} + \frac{F_l\times r - S_l}{r-1}}.$$

Else then $BF_{l-1}^{RA} \equiv F_l > S_l$, and $TR_l^{OPT} \equiv S_l$.

$$\frac{OPT[0,l]}{RA[0,l]} = \frac{\frac{BF_{l-1}^{RA}\times r}{r-1} + S_l}{\frac{BF_{l-1}^{RA}}{r-1} + BF_{l-1}^{RA}} = \frac{\frac{S_l\times r}{r-1} + \frac{BF_{l-1}^{RA}\times r - S_l}{r-1}}{\frac{S_l}{r-1} + \frac{BF_{l-1}^{RA}}{r-1} + BF_{l-1}^{RA} - \frac{S_l}{r-1}}$$

$$= \frac{\frac{S_l\times r}{r-1} + \frac{BF_{l-1}^{RA}\times r - S_l}{r-1}}{\frac{S_l}{r-1} + \frac{BF_{l-1}^{RA}\times r - S_l}{r-1}}.$$

These three equalities all satisfy the second case of the inductive hypothesis. This completes the inductive analysis. □

Using Lemmas 19, 20, and 24, we can partition the time into subintervals, and show that $OPT[i,j]/RA[i.j] \leqslant APX$ for each such subinterval $[i,j]$, proving Theorem 7.

3.3. *Lower bound for the general model*

We now briefly discuss the generalization of our lower bound construction to arbitrary queue sizes. In particular, for a queue of size B, consider the following general sequence (where $Z < B$ is some constant to be determined later):

$$\underbrace{(1,1,\ldots,1,\alpha_1)}_{B}\ \underbrace{(\alpha_1)(\alpha_1)\ldots(\alpha_1)}_{Z-1}\underbrace{(\alpha_1,\alpha_1,\ldots,\alpha_1,\alpha_2)}_{B}\ \underbrace{(\alpha_2)(\alpha_2)\ldots(\alpha_2)}_{Z-1}$$
$$\underbrace{(\alpha_2,\alpha_2,\ldots,\alpha_2,\alpha_3)}_{B}\ldots\underbrace{(\alpha_{k-1},\alpha_{k-1},\ldots,\alpha_{k-1},\alpha_k)}_{B}\ \underbrace{(\alpha_k)(\alpha_k)\ldots(\alpha_k)}_{Z-1}\underbrace{(\alpha_k,\alpha_k,\ldots,\alpha_k)}_{B}$$

If the online algorithm decides to transmit α_1 at any time $j \in [1, Z]$, then the adversary will stop the sequence at time Z, i.e., the sequence ends with $Z-1$ packets with value α_1. The online algorithm transmits no more than $(j-1) + Z\alpha_1 \leqslant (Z-1) + Z\alpha_1$, while OPT transmits all of the packets. Thus the ratio is no less than $(B-1+Z\alpha_1)/(Z-1+Z\alpha_1)$.

Otherwise, the online algorithm at best transmitted Z packets of value 1 during time $[1, Z]$. *OPT* instead would transmit Z packets of value α_1, while discarding the earlier packets with value 1. The adversary then make $B-1$ packets of value α_1 and another packet of value α_2 to arrive during time $(Z, Z+1)$, and repeats the same strategy as during times 1 through Z. If the online algorithm kept on transmitting relatively low value packets, the adversary will end the sequence with B packets with value α_k.

To maximize the lower bound, we need to balance the ratios produced in all cases, with a value Z that maximizes the overall ratio. We have the following final result:[5]

$$Z = \left\lfloor \frac{B}{2} \right\rfloor,$$

$$\alpha_i = \frac{1}{1+B}\left(\frac{2B+1+\sqrt{2B^2+2B+1}}{B}\right)^i + \frac{B}{1+B}\left(\frac{-1+\sqrt{2B^2+2B+1}}{B}\right)^i,$$

and the final bound is

$$\frac{5+\sqrt{2B^2+2B+1}}{B+4} > \frac{5+\sqrt{(\sqrt{2}B+1/\sqrt{2})^2}}{B+4} > \sqrt{2}.$$

Appendix B provides detailed calculations and so we obtain the following.

Theorem 8. *With $Z = \lfloor B/2 \rfloor$, the lower bound ratio approaches $\sqrt{2}$.*

4. Bounded delay queueing policy

In the bounded delay model, each arriving packet is also assigned a deadline. A packet must be transmitted before its deadline or else is lost. However packets can be transmitted out of order, and we assume unlimited queue sizes. We also assume deadlines are integers. The δ-uniform bounded delay model requires every packet to be transmitted within δ time units after its arrival, or otherwise it is lost. We first discuss the connection of the FIFO preemptive model to the uniform bounded delay model. In particular, we consider the FIFO preemptive model proposed in [4]. The model to be discussed in this section differs from our previous section only in that the new FIFO order permits the reordering of packets arriving at the same time unit $(j, j+1)$, for $j \in \mathbb{Z}$.

Clearly, any FIFO preemptive policy with queue size B delays each packet by at most $\delta = B$ time unit, hence also a B-uniform bounded delay policy. However, the converse is not always true. We show here that the converse is true for $B = \delta = 2$. Here we use *ANY* to denote any online 2-uniform bounded delay policy. For convenience, we round down the admitted transmission deadline to the nearest integer since transmissions only happen at integral times, i.e., a packet arriving at time t has a deadline of $\lceil t+1 \rceil$.

Theorem 9. *For $B = \delta = 2$, ANY can be modified to operate on a queue with size 2 and can serve packets in FIFO order, without degrading its performance.*

[5] These expressions are for even B. We omit the expressions for odd B for simplicity.

Proof. We modify *ANY* as follows: at any time t, first if *ANY* has more than one packet with deadline $\lceil t \rceil$, then only keep the most valuable packet. Afterwards if *ANY* still has more than two packets in its queue then we shall simply discard the packet with the least value. In case of a tie, the packet with earlier deadline is discarded. The reasoning is that *ANY* needs to transmit all the packets currently in the queue within the next two transmission times. Since only one packet can be transmitted per time unit, no more than one packet with deadline $\lceil t \rceil$ can be transmitted and in total no more than two packets can be transmitted eventually. By preserving only the two most valuable packets in the queue, we are guaranteed not to degrade *ANY*'s performance. With regard to the new FIFO ordering, at time t, the packets in *ANY*'s queue should have deadline $\lceil t \rceil$ or $\lceil t+1 \rceil$ only, since they all arrived before or at time t. This forces *ANY* to transmit the packet with deadline $\lceil t \rceil$ at time $\lceil t \rceil$, before any of the packets with later deadline $\lceil t+1 \rceil$, respecting the FIFO order. Notice that the FIFO order property only holds for $\delta = 2$. For $\delta \geqslant 3$, at any time t, we could have packets with deadlines $\lceil t+1 \rceil$ and $\lceil t+2 \rceil$ in the queue. Then it's not clear that packets with deadline $\lceil t+1 \rceil$ has to be transmitted earlier than the ones with deadline $\lceil t+2 \rceil$. *ANY* can transmit a packet with deadline $\lceil t+2 \rceil$ at time $\lceil t \rceil$, and still be able to transmit a packet with deadline $\lceil t+1 \rceil$ at time $\lceil t+1 \rceil$. In fact, we believe that the optimal online algorithm for δ-uniform bounded delay will transmit the packets out of the FIFO order for $\delta \geqslant 3$. □

The previous theorem indicates that the 2-uniform bounded delay model is equivalent to the new FIFO preemptive model with queue size 2. From now on we just focus on the new FIFO preemptive model with queue size 2, as it is closely related to the model introduced in the previous section. We first show a lower bound of 1.366 for this model. Notice that since the online algorithm can reorder the packets that arrived within the same time unit, we expect better competitive ratios for the online algorithm.

Consider the following general sequence:

$$(\alpha_0 = 1, \alpha_0 = 1)(\alpha_1)(\alpha_1, \alpha_1)(\alpha_2)(\alpha_2, \alpha_2)\ldots(\alpha_{k-1})(\alpha_{k-1}, \alpha_{k-1})(\alpha_k)(\alpha_k, \alpha_k).$$

At time 2, if the online algorithm transmits α_1, then the whole sequence ends, and the ratio is $(\alpha_0 + \alpha_0 + \alpha_1)/(\alpha_0 + \alpha_1)$. Otherwise, the online algorithm at best transmitted two α_0 packets at times 1 and 2. Now consider time 4, if the online algorithm transmits α_2, then the whole sequence ends, and the ratio becomes $(\alpha_0 + \alpha_1 + \alpha_1 + \alpha_1 + \alpha_2)/(\alpha_0 + \alpha_0 + \alpha_1 + \alpha_2)$. Otherwise, the online algorithm at best transmitted two α_2 packets at times 3 and 4. The same reasoning continues till α_k. We thus need to solve the following equations:

$$\begin{aligned}
L_k &= \frac{\alpha_0 + \alpha_0 + \alpha_1}{\alpha_0 + \alpha_1} = \frac{\alpha_0 + \alpha_1 + \alpha_1 + \alpha_1 + \alpha_2}{\alpha_0 + \alpha_0 + \alpha_1 + \alpha_2} \\
&= \frac{\alpha_0 + \alpha_1 + \alpha_1 + \alpha_2 + \alpha_2 + \alpha_2 + \alpha_3}{\alpha_0 + \alpha_0 + \alpha_1 + \alpha_1 + \alpha_2 + \alpha_3} = \cdots \\
&= \frac{\alpha_0 + \alpha_1 + \alpha_1 + \cdots + \alpha_{k-1} + \alpha_{k-1} + \alpha_{k-1} + \alpha_k}{\alpha_0 + \alpha_0 + \alpha_1 + \cdots + \alpha_{k-2} + \alpha_{k-1} + \alpha_k} \\
&= \frac{\alpha_0 + \alpha_1 + \alpha_1 + \cdots + \alpha_{k-1} + \alpha_{k-1} + \alpha_k + \alpha_k + \alpha_k}{\alpha_0 + \alpha_0 + \alpha_1 + \cdots + \alpha_{k-1} + \alpha_{k-1} + \alpha_k + \alpha_k}.
\end{aligned}$$

Using the same techniques developed in this paper, we can solve the above equations, detailed calculations are omitted. For any positive ε:

$$\alpha_i = \frac{\varepsilon}{2+2\varepsilon}\bigl(\sqrt{2+(1+\varepsilon)^2}+2+\varepsilon\bigr)^i + \left(1-\frac{\varepsilon}{2+2\varepsilon}\right)\bigl(\sqrt{2+(1+\varepsilon)^2}-\varepsilon\bigr)^i$$

and as $k \to \infty$,

$$L_k \to \frac{2+\sqrt{2+(1+\varepsilon)^2}}{1+\sqrt{2+(1+\varepsilon)^2}} > \frac{2+\sqrt{3}+\varepsilon}{1+\sqrt{3}+\varepsilon} > \frac{1+\sqrt{3}}{2} - \varepsilon \approx 1.366 - \varepsilon.$$

Theorem 10. *For FIFO queue of size* 2*, or equivalently,* 2*-uniform bounded delay queue, no deterministic online algorithm can achieve a competitive ratio of* $(1+\sqrt{3})/2 - \varepsilon \approx 1.366 - \varepsilon$*, for any constant* $\varepsilon > 0$.

Next, we introduce an online algorithm that is $\sqrt{2}$-competitive. We modify the Ratio algorithm as follows: we set the constant ratio $r = 2+\sqrt{2}$ instead, and we always reorder packets arrived in the same time unit in decreasing order. We show that the competitive ratio is bounded by

$$\max\left(\frac{1+r}{r}, \frac{r}{r-1}, \frac{3r-2}{2r-1}, \frac{r^2}{r^2-r+1}\right) = \sqrt{2} \approx 1.414$$

in Appendix C.

Theorem 11. *For FIFO queue of size* 2*, or equivalently,* 2*-uniform bounded delay queue, the Ratio algorithm achieves a competitive ratio of* $\sqrt{2} \approx 1.414$.

Next we present a matching lower bound for the 2-variable bounded delay model. The 2-variable bounded delay model allows two types of packets arriving at time t: the ones that must be sent the next integral time unit (with deadline $\lceil t \rceil$, or equivalently, delay 1), and the ones that can be delayed for 1 extra time unit (with deadline $\lceil t+1 \rceil$, or equivalently, delay 2). Consider the following general sequence. We list the associated delay of each packet in brackets after its value.

$$\bigl(1[1], \alpha_1[2]\bigr)\bigl(\alpha_1[1], \alpha_2[2]\bigr)\ldots\bigl(\alpha_{k-1}[1], \alpha_k[2]\bigr)\bigl(\alpha_k[1]\bigr).$$

The reasoning is similar as before. If at time 1, the online algorithm transmitted α_1, then the whole sequence stops, and the ratio is $(1+\alpha_1)/\alpha_1$. Otherwise, the online algorithm at best transmitted packet 1 at time 1. If at time 2 the online algorithm transmitted α_2, then again the whole sequence stops, and the ratio becomes $(\alpha_1+\alpha_1+\alpha_2)/(1+\alpha_2)$. Such reasoning continues until packet α_k. We thus need to solve the following equations:

$$\begin{aligned}
L_k &= \frac{1+\alpha_1}{\alpha_1} = \frac{\alpha_1+\alpha_1+\alpha_2}{1+\alpha_2} = \frac{\alpha_1+\alpha_2+\alpha_2+\alpha_3}{1+\alpha_1+\alpha_3} = \cdots \\
&= \frac{\alpha_1+\alpha_2+\cdots+\alpha_{k-1}+\alpha_{k-1}+\alpha_k}{1+\alpha_1+\cdots+\alpha_{k-2}+\alpha_k} \\
&= \frac{\alpha_1+\alpha_2+\cdots+\alpha_{k-1}+\alpha_k+\alpha_k}{1+\alpha_1+\cdots+\alpha_{k-2}+\alpha_{k-1}+\alpha_k}.
\end{aligned}$$

The calculation uses the same techniques before and is omitted here. For any positive ε:

$$\alpha_i = \frac{\varepsilon}{2+2\varepsilon}\left(\frac{\sqrt{4+(1+\varepsilon)^2}+3+\varepsilon}{2}\right)^i + \left(1-\frac{\varepsilon}{2+2\varepsilon}\right)\left(\frac{\sqrt{4+(1+\varepsilon)^2}+1-\varepsilon}{2}\right)^i$$

and as $k \to \infty$,

$$L_k \to \frac{1+\frac{\sqrt{4+(1+\varepsilon)^2}+1}{2}}{\frac{\sqrt{4+(1+\varepsilon)^2}+1}{2}} > \frac{3+\sqrt{5}+\varepsilon}{1+\sqrt{5}+\varepsilon} > \frac{1+\sqrt{5}}{2} - \varepsilon \approx 1.618 - \varepsilon.$$

Theorem 12. *For the* 2*-variable bounded delay model, no deterministic online algorithm can achieve a competitive ratio better than* $(\sqrt{5}+1)/2 - \varepsilon$*, for any constant* $\varepsilon > 0$.

5. Conclusion

In this paper we discussed various upper and lower bounds on competitive ratios for various queueing policies. Below we provide a summary:

- For the non-preemptive queueing policy, we proved tight $(2\alpha-1)/\alpha$ upper bound for the 2-value model, and $\Theta(\log\alpha)$ upper and lower bounds for the general value model.
- For the preemptive FIFO queueing policy, we proved 1.434 upper and lower bounds for queues of size 2, and 1.414 lower bound for general queue sizes.
- For bounded delay queueing policy, we proved 1.414 (1.366, respectively) upper (lower, respectively) bounds for the 2-uniform bounded delay model, and tight 1.618 lower bound for the 2-variable bounded delay model.

Many interesting problems remain open, for instance, are there non-trivial randomized upper and lower bounds for the preemptive and bounded delay model? For the general preemptive FIFO model, can we reduce the cap between the lower and upper bounds?

Appendix A. Solving the equations for the lower bound construction for $B = 2$

We need to solve the following two series of equalities:

$$\begin{aligned}
\frac{\alpha_0+\alpha_1}{\alpha_1} &= \frac{\alpha_0+\alpha_1}{\alpha_1},\\
\frac{\alpha_0+\alpha_1}{\alpha_1} &= \frac{\alpha_1+\alpha_1+\alpha_2}{\alpha_0+\alpha_2},\\
\frac{\alpha_0+\alpha_1}{\alpha_1} &= \frac{\alpha_1+\alpha_2+\alpha_2+\alpha_3}{\alpha_0+\alpha_1+\alpha_3},\\
\frac{\alpha_0+\alpha_1}{\alpha_1} &= \cdots,\\
\frac{\alpha_0+\alpha_1}{\alpha_1} &= \frac{\alpha_1+\alpha_2+\cdots+\alpha_{k-1}+\alpha_{k-1}+\alpha_k}{\alpha_0+\alpha_1+\cdots+\alpha_{k-2}+\alpha_k}
\end{aligned} \tag{A.1}$$

and

$$\frac{\alpha_0+\alpha_1}{\alpha_1}=\frac{\alpha_1+\alpha_2+\cdots+\alpha_{k-1}+\alpha_k+\alpha_k+\alpha_k}{\alpha_0+\alpha_1+\cdots+\alpha_{k-2}+\alpha_{k-1}+\alpha_k+\alpha_k}. \tag{A.2}$$

We manipulate equalities in (A.1) by applying the following rule: if $A/B = C/D$, then $A/B = C/D = (C - A)/(D - B)$. Here A/B and C/D represent consecutive expressions on the right hand side of (A.1). We get the following set of new equalities:

$$\begin{aligned}
\frac{\alpha_0+\alpha_1}{\alpha_1}&=\frac{\alpha_2+\alpha_1-\alpha_0}{\alpha_2-\alpha_1+\alpha_0},\\
\frac{\alpha_0+\alpha_1}{\alpha_1}&=\frac{\alpha_3+\alpha_2-\alpha_1}{\alpha_3-\alpha_2+\alpha_1},\\
\frac{\alpha_0+\alpha_1}{\alpha_1}&=\cdots,\\
\frac{\alpha_0+\alpha_1}{\alpha_1}&=\frac{\alpha_k+\alpha_{k-1}-\alpha_{k-2}}{\alpha_k-\alpha_{k-1}+\alpha_{k-2}}.
\end{aligned} \tag{A.3}$$

Keep in mind that $\alpha_0 = 1$. Solve each equation in (A.3) we have the following: $\alpha_s = (2\alpha_1 + 1) \times (\alpha_{s-1} - \alpha_{s-2})$, for $2 \leqslant s \leqslant k$. This is a linear recursion for the α_i's. The values of α_i's are determined by α_1. The series of the α_i's is a linear combination of the two geometric series: $1, p_1, (p_1)^2, \ldots, (p_1)^k$ and $1, p_2, (p_2)^2, \ldots, (p_2)^k$, where p_1 and p_2 are the two roots of the quadratic equation $x^2 = (2\alpha_1 + 1)(x - 1)$. So we have a general expression for α_i:

$$\begin{aligned}
\alpha_i&=\gamma\times(p_1)^i+(1-\gamma)\times(p_2)^i,\\
\gamma&=\frac{1}{2}-\frac{1}{2\sqrt{(2\alpha_1+1)(2\alpha_1-3)}},\\
p_1&=\frac{2\alpha_1+1+\sqrt{(2\alpha_1+1)(2\alpha_1-3)}}{2},\\
p_2&=\frac{2\alpha_1+1-\sqrt{(2\alpha_1+1)(2\alpha_1-3)}}{2}.
\end{aligned}$$

We use equality (A.2) to solve for α_1.

$$\frac{\alpha_0+\alpha_1}{\alpha_1}=\frac{\alpha_1+\alpha_2+\cdots+\alpha_{k-1}+\alpha_k+\alpha_k+\alpha_k}{\alpha_0+\alpha_1+\cdots+\alpha_{k-2}+\alpha_{k-1}+\alpha_k+\alpha_k}=\frac{\sum_{i=1}^k\alpha_i+2\alpha_k}{\sum_{i=0}^k\alpha_i+\alpha_k}\to\frac{3p_1-2}{2p_1-1}.$$

The last step is due to the following fact, which can be verified via straight forward calculation:

$$\lim_{k\to\infty}\frac{\sum_{i=0}^k\alpha_i}{\alpha_k\times\frac{p_1}{p_1-1}}=\lim_{k\to\infty}\frac{\sum_{i=1}^k\alpha_i}{\alpha_k\times\frac{p_1}{p_1-1}}=1,\quad \text{for } \gamma\neq 0.$$

Substitute in p_1 and solve the equation, we get $\alpha_1 = (1 + \sqrt{13})/2$ and in general

$$\alpha_i=\frac{1}{3}\left(\frac{\sqrt{13}+5}{2}\right)^i+\frac{2}{3}\left(\frac{\sqrt{13}-1}{2}\right)^i,$$

with $\gamma = 1/3$, $p_1 = (\sqrt{13} + 5)/2$, and $p_2 = (\sqrt{13} - 1)/2$. The final lower bound is then $(\alpha_1 + 1)/\alpha_1 = (\sqrt{13} + 5)/6 \approx 1.434$.

Appendix B. Lower bound for general queue size *B*

We need to solve the following equations (we use the notations introduced in Appendix A):

$$\begin{aligned}
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\frac{Z\alpha_1+(B-1)\alpha_1+Z\alpha_2}{Z+(Z-1)\alpha_1+Z\alpha_2},\\
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\frac{Z\alpha_1+Z\alpha_2+(B-1)\alpha_2+Z\alpha_3}{Z+Z\alpha_1+(Z-1)\alpha_2+Z\alpha_3},\\
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\cdots,\\
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\frac{Z\alpha_1+Z\alpha_2+\cdots+Z\alpha_{k-1}+(B-1)\alpha_{k-1}+Z\alpha_k}{Z+Z\alpha_1+\cdots+Z\alpha_{k-2}+(Z-1)\alpha_{k-1}+Z\alpha_k}, \qquad \text{(B.1)}\\
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\frac{Z\alpha_1+Z\alpha_2+Z\alpha_3+\cdots+Z\alpha_{k-1}+Z\alpha_k+B\alpha_k}{Z+Z\alpha_1+Z\alpha_2+\cdots+Z\alpha_{k-2}+Z\alpha_{k-1}+B\alpha_k}. \qquad \text{(B.2)}
\end{aligned}$$

We again transform the equations in (B.1) to the following:

$$\begin{aligned}
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\frac{Z\alpha_2+(B-1)(\alpha_1-\alpha_0)}{Z\alpha_2-\alpha_1+\alpha_0},\\
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\frac{Z\alpha_3+(B-1)(\alpha_2-\alpha_1)}{Z\alpha_3-\alpha_2+\alpha_1},\\
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\cdots,\\
&\frac{B-1+Z\alpha_1}{Z-1+Z\alpha_1}=\frac{Z\alpha_k+(B-1)(\alpha_{k-1}-\alpha_{k-2})}{Z\alpha_k-\alpha_{k-1}+\alpha_{k-2}}. \qquad \text{(B.3)}
\end{aligned}$$

We solve (B.3), and get the following: $(B-Z)\alpha_s=(B-1+B\alpha_1)\times(\alpha_{s-1}-\alpha_{s-2})$, for $2\leqslant s\leqslant k$. Similar to $B=2$, we can solve the sequence of α_i's using equation (B.2) in terms of Z and B. We select a Z to maximize the bound. Setting $Z=\lfloor B/2\rfloor$ gives us the following:

$$\begin{aligned}
\alpha_1&=\frac{1+\sqrt{2B^2+2B+1}}{B},\\
p_1&=\frac{2B+1+\sqrt{2B^2+2B+1}}{B},\\
p_2&=\frac{-1+\sqrt{2B^2+2B+1}}{B},\\
\gamma&=\frac{1}{1+B},\\
L_k&\to\frac{5+\sqrt{2B^2+2B+1}}{B+4}>\frac{5+\sqrt{(\sqrt{2}B+1/\sqrt{2})^2}}{B+4}>\sqrt{2}.
\end{aligned}$$

Appendix C. The Ratio algorithm is 1.414-competitive with respect to 2-uniform bounded delay model

The proof is a similar case analysis as in Section 3, here we only outline the main ideas. We inherit the notations introduced in Section 3, here *APX* denote the constant $\sqrt{2}$. First, we can conclude that *OPT* also reorder the packets arriving during the same time unit. In addition, Lemma 17 through Lemma 23 still apply in this situation. Since packets arrived during the same time unit can be reordered, we also conclude that $A_j \geqslant B_j$.

We again look for the first time where *OPT* and *RA* transmit different packets. Assume it's time 0. By Lemma 19, we know $F_0 < S_0$. There are again two cases, we will consider them separately.

The first case is that $TR_0^{RA} \equiv S_0$, while $TR_0^{OPT} \equiv F_0$. Since $S_0 > F_0$, $TR_1^{OPT} \equiv BF_0^{OPT} \equiv S_0$. By the definition of *RA*, $S_0 \geqslant r \times F_0$. Thus

$$\frac{OPT[0,1]}{RA[0,0]} = \frac{F_0 + S_0}{S_0} \leqslant \frac{F_0 + r \times F_0}{r \times F_0} \leqslant \frac{1+r}{r} \leqslant APX.$$

The second case is more involved, when $TR_0^{RA} \equiv F_0$ and $TR_0^{OPT} \equiv S_0$. We reproof Lemma 24 below.

Proof. The base case where $j = 0$: if $BF_0^{OPT} \neq 0$, then $BF_0^{OPT} \leqslant F_0$. Then either $TR_1^{RA} \equiv S_0$, in which case

$$\frac{OPT[0,1]}{RA[0,1]} \leqslant \frac{S_0 + F_0}{F_0 + S_0} = 1 < APX.$$

Or $TR_1^{RA} \not\equiv S_0$, so $TR_1^{RA} \geqslant S_0 \times r$ and S_0 was thrown away. Then $TR_2^{OPT} \equiv BF_1^{OPT} \equiv TR_1^{RA}$, in which case

$$\frac{OPT[0,2]}{RA[0,1]} \leqslant \frac{S_0 + F_0 + TR_1^{RA}}{F_0 + TR_1^{RA}} \leqslant \frac{S_0 + F_0 + TR_1^{RA}}{F_0 + S_0 + TR_1^{RA}\frac{r-1}{r}} < \frac{r}{r-1} \leqslant APX.$$

Else $BF_0^{OPT} = 0$. By the definition of *RA*, $F_0 > S_0/r$, then:

$$\frac{OPT[0,0]}{RA[0,0]} = \frac{S_0}{F_0} = \frac{\frac{S_0 \times r}{r-1} - \frac{S_0}{r-1} + 0}{\frac{S_0}{r-1} - \frac{S_0}{r(r-1)} + \left(F_0 - \frac{S_0}{r}\right)},$$

where

$$\frac{\frac{S_0}{r-1}}{\frac{S_0}{r(r-1)}} = r > APX > \frac{0}{F_0 - \frac{S_0}{r}}.$$

This satisfies the inductive hypothesis, completing the base case analysis for $j = 0$.

Assume now at time $l - 1$ the inductive hypothesis is true, consider time l. As in Lemma 24, the interesting case is that the second condition of the inductive hypothesis held at time $l - 1$ and some new packets arrived during time $(l - 1, l)$. Here are the possibilities:

(1) If $TR_l^{RA} \equiv S_l$. Then $S_l/F_l \geqslant r$, and $F_l \equiv BF_{l-1}^{RA}$. Then $TR_l^{OPT} \equiv S_l$, since $BF_{l-1}^{OPT} = 0$, implying:

$$\frac{OPT[0,l]}{RA[0,l]} \leqslant \frac{\frac{BF_{l-1} \times r}{r-1} + S_l}{\frac{BF_{l-1}}{r-1} + S_l} = \frac{r^2}{r^2 - r + 1} \leqslant APX.$$

(2) Else then $TR_l^{RA} \equiv F_l$ and $BF_l^{RA} \equiv S_l$. If $BF_l^{OPT} \neq 0$, the analysis for this part is identical to that of Lemma 24, where we obtain two bounds:

$$\frac{3r-2}{2r-1} \leqslant APX \quad \text{and} \quad \frac{r}{r-1} \leqslant APX.$$

(3) Else then $TR_l^{RA} \equiv F_l$ and $BF_l^{RA} \equiv S_l$, and $BF_l^{OPT} = 0$. The analysis for this part is again identical to that of Lemma 24. □

This completes the proof for Theorem 11.

References

[1] W.A. Aiello, Y. Mansour, S. Rajagopolan, A. Rosen, Competitive queue policies for differentiated services, in: Proceedings of the IEEE INFOCOM, 2000, pp. 431–440.
[2] D. Clark, J. Wroclawski, An approach to service allocation in the internet, Internet draft, available from http://diffserv.lcs.mit.edu, 1997.
[3] C. Dovrolis, D. Stiliadis, P. Ramanathan, Proportional differentiated services: delay differentiation and packet scheduling, in: Proceedings of ACM SIGCOMM, 1999, pp. 109–120.
[4] A. Kesselman, Z. Lotker, Y. Mansour, B. Patt-Shamir, B. Schieber, M. Sviridenko, Buffer overflow management in QoS switches, in: Proceedings of ACM STOC, 2001, pp. 520–529.
[5] A. Kesselman, Y. Mansour, Loss-bounded analysis for differentiated services, in: Proceedings of SIAM–ACM SODA, 2001, pp. 591–600.
[6] A. Kesselman, Y. Mansour, R. Van-Stee, Improved competitive guarantees for QoS buffering, in: Proceedings of ESA, 2003, pp. 361–372.
[7] T. Nandagopal, N. Venkitaraman, R. Sivakumar, V. Bharghavan, Relative delay differentiation and delay class adaptation in core-stateless networks, in: Proceedings of IEEE INFOCOM, 2000, pp. 421–430.
[8] N. Semret, R.R.-F. Liao, A.T. Campbell, A.A. Lazar, Peering and provisioning of differentiated internet services, in: Proceedings of IEEE INFOCOM, 2000, pp. 414–420.
[9] I. Stoica, H. Zhang, Providing guaranteed services without per flow management, in: Proceedings of ACM SIGCOMM, 1999, pp. 81–94.
[10] J.S. Turner, New directions in communications, IEEE Commun. Magazine 24 (1986) 8–15.